SFB 649 Discussion Paper 2017-008

# GitHub API based QuantNet Mining infrastructure in R

Lukas Borke*
Wolfgang K. Härdle*

*Humboldt-Universität zu Berlin, Germany

This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 "Economic Risk".

http://sfb649.wiwi.hu-berlin.de
ISSN 1860-5664

SFB 649, Humboldt-Universität zu Berlin
Spandauer Straße 1, D-10178 Berlin

# GitHub API based QuantNet Mining infrastructure in R*

Lukas Borke† Wolfgang K. Härdle‡

**Abstract**

QuantNet being an online GitHub based organization is an integrated environment consisting of different types of statistics-related documents and program codes called Quantlets. The QuantNet Style Guide and the **yamldebugger** package allow a standardized audit and validation of YAML annotated software repositories within this organization. The behavior statistics of QuantNet users are measured with Web Metrics from **Google Analytics**. We show how the search queries obtained from Google's metrics can be used in the *test collections* in order to calibrate and evaluate the information retrieval (IR) performance of QuantNet's search engine called QuantNetXploRer. For that purpose, different text mining (TM) models will be examined by means of the new **TManalyzer** package. Further, we introduce the ***Validation Pipeline*** (*Vali-PP*) and apply it on the YAML data. *Vali-PP* is a functional multi-staged instrument for clustering analysis, providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the pipeline. The new package **rgithubS**, which enables a GitHub wide search for code and repositories using the GitHub Search API and which is an essential element of the QuantNet Mining infrastructure, is briefly presented.

The **TManalyzer** results show that for all considered single term queries the number of true positives is maximal in a latent semantic analysis model configuration (LSA50). The *Vali-PP* analysis indicates that the optimality of the combination LSA50 and hierarchical clustering (HC) applies to $70 - 90\%$ of the cluster sizes for most of the considered quality indices. Further, we can infer that more accurate and comprehensive metadata increases the clustering quality. Subsequently, the findings of our experimental design are implemented into the QuantNetXploRer. The GitHub API driven QuantNetXploRer can be found and mined under http://www.quantlet.de

*Keywords*: Code Search, Software Repositories, Text Mining, Information Retrieval, Smart Data, YAML, GitHub Search API, Google Analytics, Web Metrics, LSA, GVSM, Cluster Validation, Quality Indices, Validation Pipeline

*We acknowledge financial support for this project from the Deutsche Forschungsgemeinschaft (DFG) through CRC 649 Economic Risk.

†Humboldt-Universität zu Berlin, R.D.C - Research Data Center, SFB 649 "Economic Risk", Spandauer Str. 1, 10178 Berlin, Germany

‡Humboldt-Universität zu Berlin, C.A.S.E. - Center for Applied Statistics and Economics, Unter den Linden 6, 10099 Berlin, Germany and School of Business, Singapore Management University, 50 Stamford Road, Singapore 178899

# 1 Introduction to GitHub Mining

This research is the technological and scientific basis of the project "GitHub API based QuantNet Mining infrastructure in R" as introduced in Borke and Härdle (2017). Its structure and objectives are described by the diagram in Figure 1 and in the following text. The research starts with the *Parser 1* node of the TM Pipeline (see Figure 1) and goes along the path till the end point at the *Smart Clusterization* node. Alternatively, the pipeline could start with other parsers, depending on the data source. For instance, *Parsers* 2 or 3 could be used for processing the special repository structure of papers or external books, respectively.

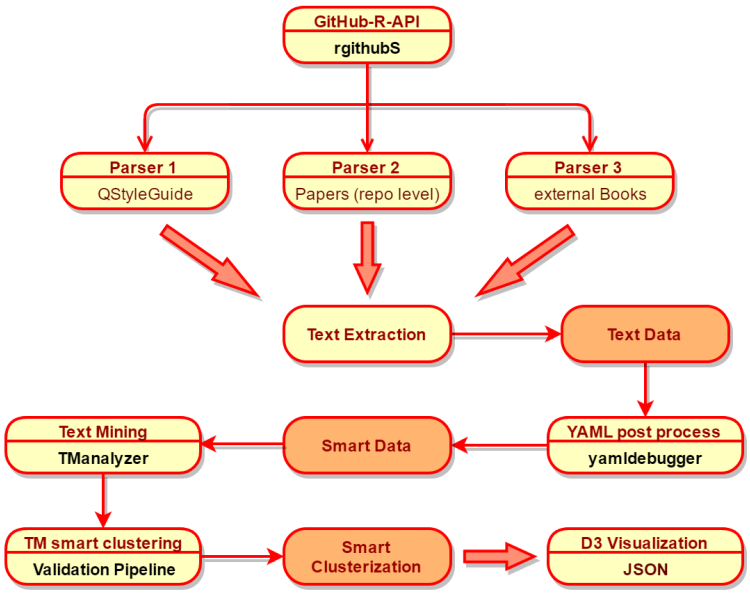


Figure 1: TM Pipeline of the "GitHub API based QuantNet Mining infrastructure in R"

An integral part of the overall project is **GitHub** – a Git repository hosting service founded in 2008, which not only provides its users with a web-based graphical interface of the well-known version control system, but also allows such high-level features as access control and collaborative work (Loeliger, 2009). The first essential step of the large scheme is to adjust the application program interface (API) of GitHub to the R software environment and to implement different parsers for data extraction from various repositories of programming projects, see the **rgithubS** package (Scheidegger and Borke, 2017), whose baseline operational spectrum is described in Borke and Bykovskaya (2017). Big Data obtained at this stage is thus constituted by a large corpus of text documents, each of which corresponds to the meta information of a particular program code, be it in a repository, a folder or under another GitHub storage resource.

QuantNet was originally designed as a platform to freely exchange empirical as well as quantitative-theoretical methods for statistical and economical programming, called Quantlets, or in abbreviated form "QLs". It supports the deployment of computer codes written in R, Matlab, SAS and Python. Because of the open structure other languages can be easily added.

**The first objective** is to implement initial processing of the massive text data obtained

from the QuantNet's GitHub organization, available at `https://github.com/Quantlet`. In the case of QuantNet, the task of extracting smart data out of the raw text collection is completed by means of the **rgithubS** and the **yamldebugger** packages (Borke, 2017b), using the YAML (`http://yaml.org/`) encoded metadata of the QLs, see Sections 3 and 4. Being a human-readable data serialization language, YAML is commonly used for configuration files, but could be used in many applications where data is being stored (e.g. debugging output) or transmitted (e.g. document headers). Thus derived smart data pass through a compound chain of processing layers, TM and Smart Clustering layer. The TM layer is implemented in form of the **TManalyzer** package (Borke, 2017a), see Section 6. The Smart Clustering layer is calibrated via the **Validation Pipeline** (*Vali-PP*) as described in Section 7.

**The second objective** is to calibrate the IR performance and effectiveness in different TM models. Section 5 shows how the search queries obtained from Web Metrics via the **RGoogleAnalytics** package (Pearmain et al., 2014) can be exploited for this purpose. Further, different clustering and validation methods within the R software environment are examined in order to determine the optimal combination of the data configuration, vector space model, clustering method and clustering criteria settings. The *Vali-PP* evaluates thereby the resulting partition and eventually finds a reasonable number of clusters, see Section 7.1.1 for more details. While Section 2 provides the related work of this research, Section 8 presents an overview of the findings.

In a summary, the GitHub-R-API based and **rgithubS** driven TM pipeline (including three parser types as displayed in Figure 1) retrieves the YAML encoded meta information of Quantlets via the **yamldebugger** package, then the LSA model is applied, clusters and labels are generated (by use of the **TManalyzer** package and **Validation Pipeline**) and the processed data is transferred via JSON into the D3 application, which is the visualization layer of the QuantNetXploRer.

# 2 Related Work

D3.js (or just D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. The QuantNetXploRer is a good example of D3 in power. More information about the D3 architecture, its various designs and the D3-based QuantNetXploRer can be found in Bostock et al. (2011) and Borke and Härdle (2017). The repository `https://github.com/Quantlet/D3Genesis` contains detailed information about the development of the main D3 components for the QuantNet visualization together with live examples on GitHub pages.

One of studies presenting the effectiveness of LSA was performed by Feinerer and Wild (2007). They applied LSA based algorithms for the automated processing of transcripts of interviews. Compared to marketing expert judgments, the machine results showed very high levels of reliability and validity in automatic text analysis. Moreover, the LSA approach proved useful not only in avoiding human inherent subjectivity, but also in reducing high costs of human judgment.

Wild and Stahl (2007) described the **lsa** R package (Wild, 2015) and illustrated its proper

use through examples from the areas of automated essay scoring and knowledge representation. Feinerer et al. (2008) presented the **tm** package (Feinerer and Hornik, 2015) which provides a framework for text mining applications within R, encompassing techniques for count-based analysis methods, text clustering, text classification and string kernels. The new package **TManalyzer** (Borke, 2017a) combines and extends the functionality of both packages, facilitating IR tools in 3 text mining models: BVSM, GVSM(TT) and LSA. As presented in Cristianini et al. (2002) and Borke and Härdle (2017), all three TM models are special representations of the generalized vector space model (GVSM).

Brock et al. (2008), Desgraupes (2013) and Charrad et al. (2014) provided a good overview about the existing clustering validity indices. Additionally, there are accompanying R packages for their application allowing cluster validation and determining the relevant number of clusters in a data set: **clValid**, **clustCrit**, **NbClust**.

Gousios and Spinellis (2012) performed a deep analysis on the architecture of GitHub data and provided the overall schema of GitHub's data and API, discussing ways to overcome its limitations. Their paper also presents GHTorrent, an effort to create a scalable, queriable, offline mirror of data offered through the GitHub REST API, providing users with a possibility to analyze the development of their projects, and researchers with efficient tool to gather and analyze GitHub's event-stream data. Based on GHTorrent data, Kalliamvakou et al. (2014) analyzed the quality and properties of the data available from GitHub and discussed both positive and negative points of using and mining GitHub.

Scheidegger and his co-authors (North et al., 2015) introduced the design and implementation of the RCloud[1], an Integrated Exploratory Analysis, Visualization, and Deployment on the Web. Being an environment for collaboratively creating and sharing data analysis scripts, RCloud encompasses analysis code in R, HTML5, Markdown, Python, and others. Amongst other features, RCloud provides an environment, in which R packages can create rich HTML content, using, for example, D3 and dc.js, and a transparent, integrated version control system. RCloud's implementation[2] of the versioning mechanism is built on top of GitHub's gists[3]. The **github** package (Scheidegger, 2016) supports, amongst many other features, creating, modifying and administrating of GitHub's gists via the GitHub API.

The new package **rgithubS** (Scheidegger and Borke, 2017), which extends the functionality of the **github** package, allows a GitHub wide search for code and repositories using the GitHub Search API. Performing similar as Google, it is designed to find results that best meet the personal needs and which are ranked by best match, as indicated by the score field for each item returned. Borke and Bykovskaya (2017) introduce the QuantNet@GitHub statistics and some lightweight parsers within the baseline operational spectrum of **rgithubS**. The current "QuantNet@GitHub" statistics are retrieved in real time as displayed in Table 1 (on February 28, 2017). First we see there the total number of all QLs on GitHub, then all QLs in the `https://github.com/Quantlet` organization. In the third place, all Style Guide compliant and validated QLs from the QuantNetXploRer visualization are displayed (`http://quantlet.de/`). Additionally, an optional character vector as an argument for the desired author/editor list is supported.

[1] `https://github.com/att/rcloud`

[2] `http://rcloud.social/gallery/index.html`

[3] `https://help.github.com/articles/about-gists/`

| | total number |
|---|---|
| full_gh | 2799 |
| quantlet_gh_org | 1495 |
| QuantNetXploRer | 1239 |
| bykovskaya.as.editor | 203 |
| borke.as.editor | 149 |

Table 1: QuantNet@GitHub statistics via the *qnet.stats* function from the **rgithubS** package

# 3 QuantNet Search Code in a nutshell

Borke and Bykovskaya (2017) show how the GitHub Search API (`https://developer.github.com/v3/search/`) can be used for software mining of QuantNet and other GitHub organizations. Just like searching on Google, people want to see a few pages of search results so that one can find the item that best meets the personal needs. To satisfy that need, the GitHub Search API provides up to 1.000 results for each search. GitHub's results are sorted by best match, as indicated by the score field for each item returned.

```
library(rgithubS)
library(yamldebugger)
# GitHub's user authorization
ctx = interactive.login("client_id", "client_secret")

q_search = 'Quantlet Published Description Keywords Author filename:"metainfo.txt"'

spec_search_term = "yaml user:Quantlet user:lborke user:b2net"
sr = search.code(paste(spec_search_term, q_search), per_page = 20)

spec_search_term = "black scholes user:Quantlet"
sr = search.code(paste(spec_search_term, q_search), per_page = 10)

sr$content$total_count

q_top = yaml.parser.light(sr, print_item = FALSE)

(q_names  = sapply( q_top, function(yaml_meta){ yaml.getQField(yaml_meta, "q")} ))
(q_author = sapply( q_top, function(yaml_meta){ yaml.getQField(yaml_meta, "a")} ))
(q_scores = sapply( sr$content$items, function(item){ item$score} ))
(q_repos  = sapply( sr$content$items, function(item){ item$repository$full_name }))

( name_path_scores = data.frame(qlet.name = q_names, qlet.repo.path = q_repos,
                                search.score = round(q_scores, 2)) )

a_splitted = unlist(str_split(q_author, ", "))
( tab_sorted = sort(table(a_splitted), decreasing = T)[1:10] )

( q_s = qnet.stats(spec_editor = c("bykovskaya", "borke")) )
```

Listing 1: QuantNet Search Code via the **rgithubS** package

Listing 1 produces the results for Tables 1, 2, 3 and 4. After loading the package **rgithubS** and "Basic Authentication"[4] the *Search code* functionality of the GitHub Search API is

[4] `https://developer.github.com/v3/search/#rate-limit`

fully available. Two search queries are performed by the function `search.code`. The first search query retrieves all YAML meta infos according to the Style Guide and containing the term "yaml" from 3 users/organizations: Quantlet, lborke, b2net (see Table 2). Due to the parameter `per_page = 20` only the top twenty matches (sorted by the score value) are retrieved. These results are presented in Table 2. The second query specifies all meta infos in the organization `https://github.com/Quantlet`, which share the term "black scholes", see Table 3. As `per_page` was set to 10, only the top ten results are collected. Additionally, the variable `sr$content$total_count` provides the total count of all matches, 98 QLs in this case. The top ten authors (concerning the number of contributions) of the QLs retrieved by the second query are presented in Table 4. For that purpose `per_page` was set to 100 in order to capture all relevant QLs. Via the function `qnet.stats` the current "QuantNet@GitHub" statistics are retrieved in real time as displayed in Table 1.

| | qlet.name | qlet.repo.path | score |
|---|---|---|---|
| 1 | yaml_run | lborke/yamldebugger_intro | 12.05 |
| 2 | yaml_keyword_finder | lborke/yamldebugger_intro | 11.97 |
| 3 | yaml_start | lborke/yamldebugger_intro | 11.71 |
| 4 | YAMLcentroids | b2net/Clustering_Validation_Pipeline | 11.70 |
| 5 | YAMLcleanmerge | b2net/Clustering_Validation_Pipeline | 11.70 |
| 6 | YAMLnumbchars | b2net/Clustering_Validation_Pipeline | 11.70 |
| 7 | lsa_heatmaperr | b2net/Clustering_Validation_Pipeline | 11.70 |
| 8 | lsa_heatmapsvd | b2net/Clustering_Validation_Pipeline | 11.70 |
| 9 | yaml_wordcloud | lborke/yamldebugger_intro | 11.67 |
| 10 | yaml_keyword_frequency | lborke/yamldebugger_intro | 11.65 |
| 11 | lsa_determineSign | b2net/Clustering_Validation_Pipeline | 11.50 |
| 12 | yaml_TDM_CorrPlot | lborke/yamldebugger_intro | 11.42 |

Table 2: All Quantlets dealing with "YAML" available on Quantlet, lborke, b2net; extracted via **rgithubS** and **yamldebugger**

| | qlet.name | qlet.repo.path | search.score |
|---|---|---|---|
| 1 | blspricevec | QuantLet/SFS-ToDo | 11.98 |
| 2 | SFSstoploss | QuantLet/SFS | 11.09 |
| 3 | blsprice | QuantLet/SFE-ToDo | 11.09 |
| 4 | SFEItoProcess | QuantLet/SFE_class_2015 | 10.90 |
| 5 | SFEBoundary | QuantLet/SFE_class_2015 | 10.90 |
| 6 | SFEBoundary_V | QuantLet/SFE_class_2015 | 10.90 |
| 7 | SFEBoundary_V_tau | QuantLet/SFE_class_2015 | 10.90 |
| 8 | blackscholes | QuantLet/SFS | 10.81 |
| 9 | SFSBSCopt1 | QuantLet/SFS | 10.81 |
| 10 | SFShullhedgeratio | QuantLet/SFS | 10.76 |

Table 3: Top ten Quantlets from the Quantlet organization dealing with "black scholes", extracted via **rgithubS** and **yamldebugger**

For the aggregation of the search results as displayed in the tables of this section, an additional parsing process of the retrieved YAML meta infos (whose locations are stored in the data structure `sr$content$items`) is necessary. This is accomplished via the

lightweight parser `yaml.parser.light`, which is a function and part of the **rgithubS** package. The `yaml.parser.light` produces a list of parsed YAML objects, for each YAML file one "YAML list object" with YAML data fields as further list elements. Hence, the resulting list object `q_top` contains each of the 12 "YAML list objects" (in the case of the search string "yaml"). The package **yamldebugger** for extracting the YAML data fields (`yaml.getQField`) is incorporated. Finally, all information is merged into the corresponding data frames for table creation (e.g. `name_path_scores`).

| | Number of Quantlets |
|---|---|
| Awdesch Melzer | 41 |
| Ying Chen | 14 |
| Andreas Golle | 12 |
| Christian M. Hafner | 6 |
| Lasse Groth | 6 |
| Szymon Borak | 6 |
| Florian Schulz | 5 |
| Simon Gstöhl | 5 |
| Daniel T. Pele | 4 |
| Derrick Kanngiesser | 4 |

Table 4: Top ten authors of Quantlets dealing with "black scholes", extracted via **rgithubS** and **yamldebugger**

More application examples of the **rgithubS** package for mining different types of GitHub organizations are illustrated in Borke and Bykovskaya (2017).

# 4 Yamldebugger

## 4.1 YAML

YAML is a human friendly data serialization standard for all programming languages (`http://yaml.org/`). Designed as a human-readable and data-oriented language in 2001, YAML can easily be applied to widely used data frames such as lists and arrays. What makes YAML also rather user-friendly for maintaining hierarchical data is that it avoids the excessive use of brackets, tags and other enclosures which could make the document structure less comprehensible.

The design goals for YAML are, in decreasing priority: 1. YAML is easily readable by humans; 2. YAML data is portable between programming languages; 3. YAML matches the native data structures of agile languages; 4. YAML has a consistent model to support generic tools; 5. YAML supports one-pass processing; 6. YAML is expressive and extensible; 7. YAML is easy to implement and use.

There exist many YAML parser implementations for various programming languages, amongst them: C/C++, Java, Javascript, PHP, Python, Ruby. The R implementation is available as a package (`https://github.com/cran/yaml`), which is basically a C interface to the 'libyaml', a YAML 1.1 parser and emitter, see `http://pyyaml.org/wiki/LibYAML`.

Due to the properties mentioned above, YAML was selected as annotation language for the meta information of QLs. A typical example for a YAML meta info is e.g.: `https://github.com/Quantlet/QuachSymanzikForsgren/blob/master/Metainfo.txt`. Besides Quantlet, there are various GitHub organizations using YAML for metadata. Three examples with each more than 30.000 repositories are:
I) `https://github.com/GITenberg` – an open source community curating and publishing highly usable and attractive ebooks in the public domain stored as a collaborative, trackable, scriptable digital library on GitHub;
II) `https://github.com/gitpan` – a project to import the entire history of CPAN (Comprehensive Perl Archive Network) into a set of git repositories, one per distribution; and
III) `https://github.com/the-domains` – a big collection of meta information concerning web pages and their images.

## 4.2 Style Guide

The QuantNet Style Guide[5] enables a standardized audit and validation of new QLs by means of comprehensive help pages and the **yamldebugger** package, see also Section 4.3.

The Style Guide contains several subsections:
1) Style guide of Quantlets: an overview of the structure of a Quantlet;
2) Characteristics and mandatory data fields of the YAML meta info file *Metainfo.txt*;
3) Examples of complete and correct meta infos;
4) The main YAML rules most relevant for QuantNet;
5) Instructions on how to format the programming R code with examples of using the **formatR** package (Xie, 2016a);
6) Basic instructions for the GitHub Desktop client (`https://desktop.github.com/`);
7) Main information about the purpose of the **yamldebugger** package and further guidelines (technical terms, Quantlet repository structure, special characters etc.).

The QuantNet Style Guide was developed by several Quantlet users[6] over a longer time period and was permanently adjusted to the practical needs. Together with the **yamldebugger** and introductory yamldebugger_intro, a potential Quantlet contributor has all necessary tools for a fast, transparent and iterative code development and documentation process.

## 4.3 Yamldebugger package

In order to simplify and automate the validation process of new QLs, the YAML parser debugger package (or **yamldebugger** for short) (Borke, 2017b) was developed for testing and certifying of local versions of the GitHub repositories containing YAML metadata, see `https://github.com/Quantlet/yamldebugger` for implementation details. The yamldebugger fulfills **two main tasks**. First, it checks the Quantlet repository structure, the

[5] `https://github.com/Quantlet/Styleguide-and-FAQ`
[6] `https://github.com/Quantlet/Styleguide-and-FAQ/graphs/contributors`

validity of the YAML meta information and the completeness of the mandatory data fields as described in the Style Guide, see Section 4.2. Second, the yamldebugger helps to analyze, standardize and unify the different YAML data fields, which are subject to varying spelling and notations.

The current yamldebugger version ranks every validated QL, thus helping to quickly identify deviations and discrepancies from the Style Guide specifications as well as YAML errors. The quality ranking system spans five different grades: "A", "B", "C", "D" and "N". "A" means the full compliance, "B" minor discrepancies, "C" more serious style violations and "D" YAML parser errors. "N" indicates that no YAML meta info could be found and must be decided on an individual basis, because a repository can contain different subfolders, those containing QLs and those without them.

```
library(yamldebugger)

workdir = "C:/GitHub/Stochastic_processes"
d_init = yaml.debugger.init(workdir, show_keywords = FALSE)
qnames = yaml.debugger.get.qnames(d_init$RootPath)
d_results  = yaml.debugger.run(qnames, d_init)
( OverView = yaml.debugger.summary(qnames, d_results, summaryType = "compact") )
```

Listing 2: The interaction of the four main functions of the **yamldebugger** package

Listing 2 demonstrates the interaction of the four main functions of the **yamldebugger**. This set of functions is responsible for the first main task (validity of the YAML meta information and repository structure). Listing 11 demonstrates the practical application, using the repository `https://github.com/Quantlet/Stochastic_processes` as an example.

```
subset(OverView, !(`Q-Quali` %in% c("A")) )
yaml.not.Qdfields(d_results$meta_names_distribution)
rowSums(sapply( d_results$Metainfos, function(yaml)
              { yaml.Qdfields.nchar.from.meta(yaml) } ))
d_names = unlist(sapply( d_results$Metainfos, function(yaml)
              { yaml.Qdfields.from.meta(yaml)$found_dnames } ))
( d_names_distr = sort(table(d_names), decreasing = TRUE) )
```

Listing 3: YAML data field analysis via **yamldebugger** functions

Listing 3, on the other hand, illustrates the interplay of the **yamldebugger** functions for YAML data field analysis. The corresponding example is given in Listing 12 using the same QLs for validation as in Listing 11. Diverse characteristics as quality ranking grades, distributions and occurrences of data field names, their validity, their character distributions etc. can be aggregated, analyzed, evaluated, and, if necessary, the YAML data field matching list `Q_dfield_list`[7] from the package itself can be adjusted. A meaningful and reasonable calibration of this matching list is crucial for further extraction of YAML data fields (via the `yaml.getQField` function of the **yamldebugger** package) within the TM and cluster validation steps, as will become apparent in Sections 6.3 and 7.1.2. The function `yaml.getQField` was already used in Listing 1.

The function `yaml.debugger.summary` in Listing 2 allows three different levels of summary details: `mini`, `compact` and `full`. Together with the R package **knitr** (Xie, 2016b),

[7] `https://github.com/lborke/yamldebugger/blob/master/R/yaml.Qdfields.R`

the yamldebugger summary can be easily converted into a GitHub compliant Markdown table via `kable(OverView)`. Since the **yamldebugger** version 1.0 all validated Quantlet repositories contain the file "yamldebugger_results.md", see e.g. the SFE repository[8]. This reporting process can be even simplified by means of the package **git2r** (Widgren and others, 2016).

The introductory QLs yamldebugger_intro provide more examples on how to install and run the **yamldebugger** with additional analysis and visualization capabilities, see also Figures 2 and 10. The yaml_TDM_CorrPlot visualizes correlations between the most frequent keywords of the document-term matrix, which is extracted from the keywords in the Quantlet YAML meta infos, see Figure 2. Listing 4 shows the relevant code part from the yaml_TDM_CorrPlot.

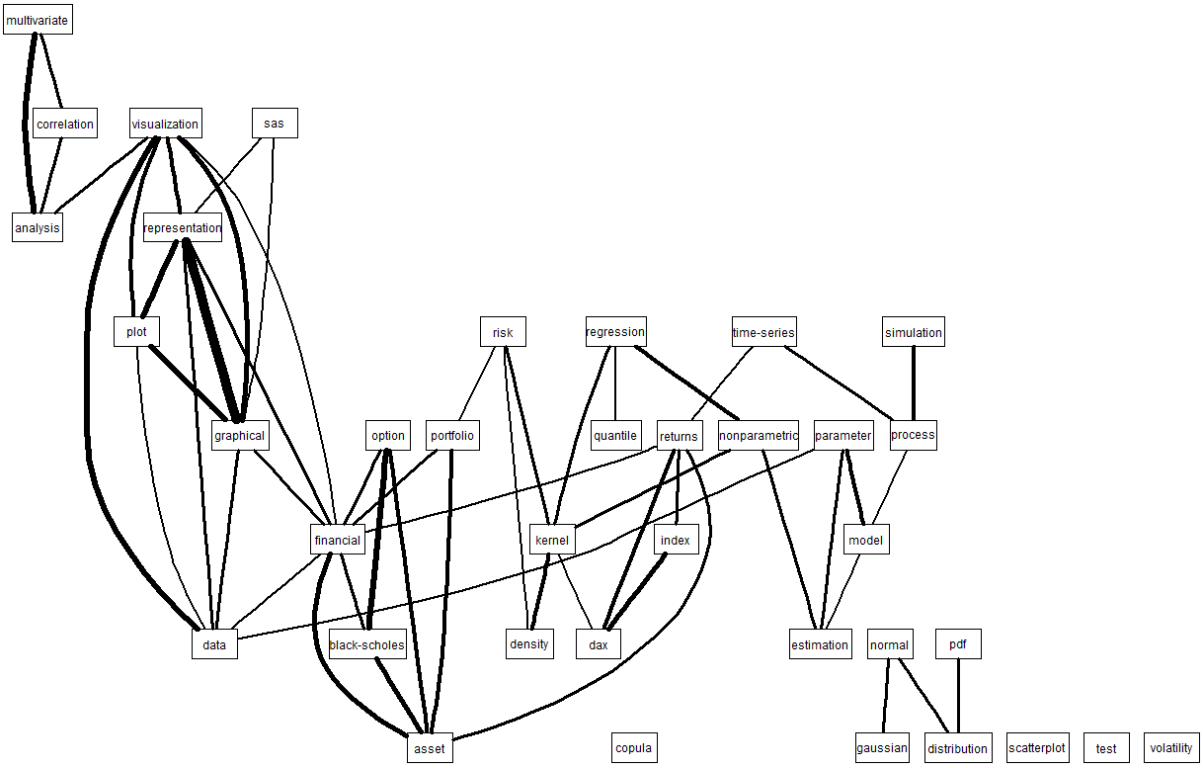


Figure 2: Correlation plot of YAML keywords

```
> DTM
<<DocumentTermMatrix (documents: 1198, terms: 980)>>
Non-/sparse entries: 11383/1162657
Sparsity           : 99%
Maximal term length: 35
Weighting          : term frequency (tf)
> DTM_graph = plot(DTM, terms = findFreqTerms(DTM, lowfreq = 60), corThreshold = 0.2,
    weighting = TRUE)
> DTM_graph
[1] "A graph with 37 nodes."
```

Listing 4: yaml_TDM_CorrPlot application example

[8] https://github.com/Quantlet/SFE/blob/master/yamldebugger_results.md

# 5 Google Analytics

The aim of this section is to provide insights into the automatized integration of data from *Google Analytics* into R for further processing and analysis. The data of interest are various download statistics of the QuantNet website. Within the *Collaborative Research Center 649: Economic Risk*, there was a regular meeting in which a certain set of statistics was presented. These presentations were created on a monthly basis and in a specific build-up. Their content however needed to be updated each time based on the log files of the Linux server. Originally, this was done manually in *Excel* and then exported to *PowerPoint*. Later, this was adopted into the R environment by means of the *Google Analytics API*, such that up-to-date statistics are available through a program in R.

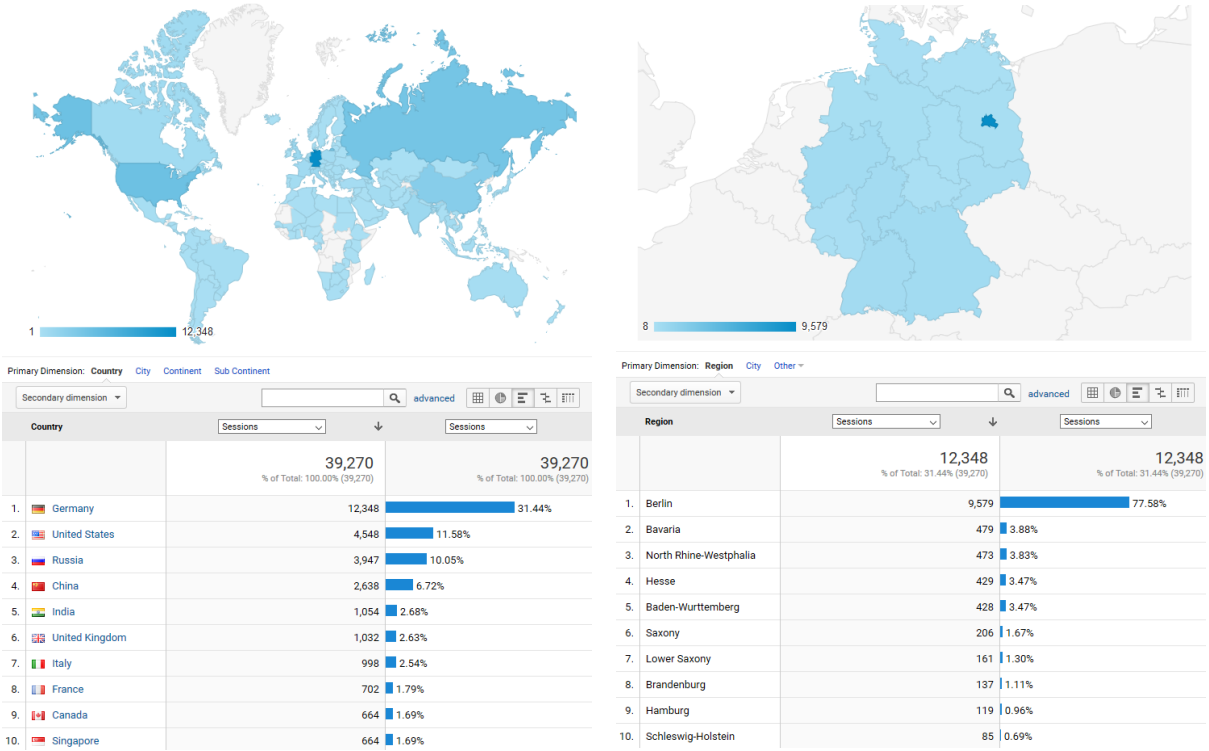


Figure 3: QuantNet visitors via *Google Analytics*: global view (left), Germany(right)

## 5.1 Introduction to Web Metrics

*Web metrics*, also known as *web analytics*, is the process of collecting, analysing and reporting online traffic generated by internet users on a website. This can be helpful for improving the usability of a particular website or to get valuable information about the relevance of the provided content. There are several tools to analyse the web traffic which differ in their functionalities and complexity. In a nutshell, these tools cover combinations of browser logging tools and user panels, the collection of network traffic data provided via the Internet service provider as well as site-specific server log parsers or page tagging technologies.

*Google Analytics* represents the latter. It is a page-tagging tool, which employs first party cookies to track user behavior. This means, that user data is collected via the web

browser and sent to a remote data-collection server. Google provides this service and the relevant reports for further analysis (`https://www.google.com/analytics/`). Kaushik (2010) delivers a good introduction into the field of online data mining and *predictive analytics*. Prem et al. (2016) present various indicators (among them *Google Analytics*) to measure open science implementations and to create an Open Science Observatory (`http://opendigitalscience.eu`).

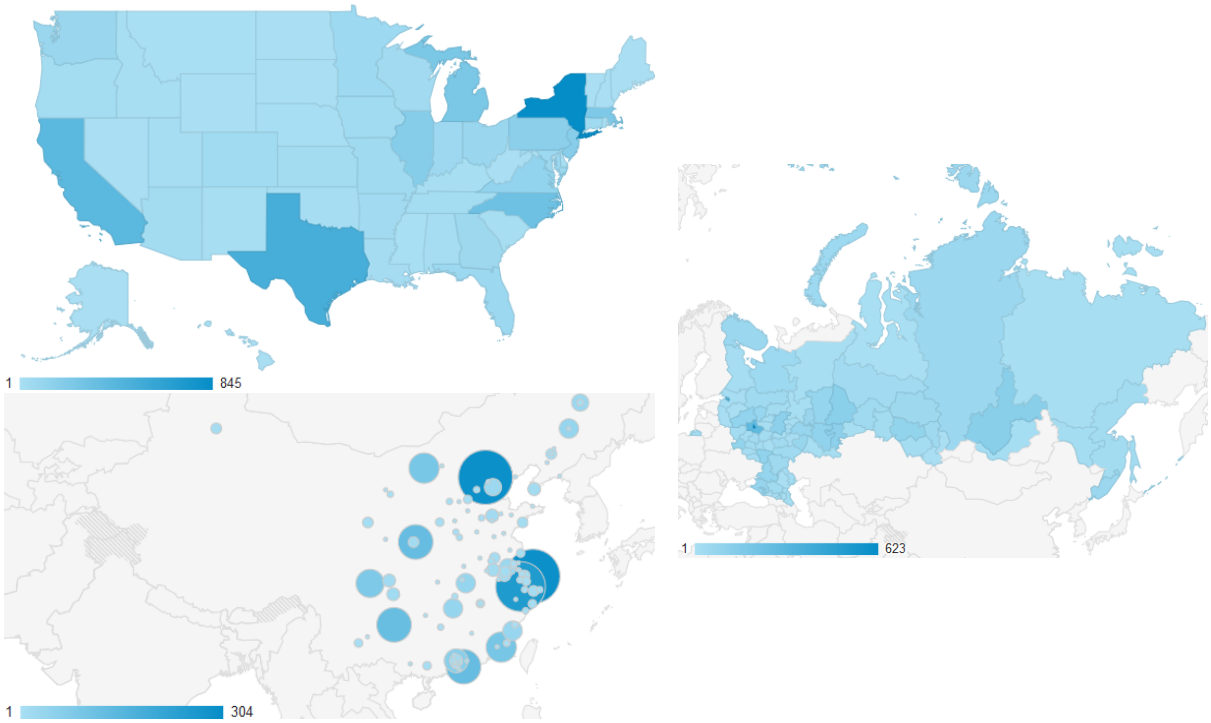


Figure 4: QuantNet visitors from USA (upper left), Russia (right) and China (lower left)

Figures 3 and 4 show the *Audience Overview*, which is accessible via the *Google Analytics* website. This reporting tool allows manual configuration and parametrization of the desired web analytics results. In the given case, the total user sessions in the time period 16.11.2013 – 18.11.2016 are grouped and sorted by countries. The most visitors come from Germany (Figure 3), followed by those from the USA, Russia and China (Figure 4).

## 5.2 RGoogleAnalytics in a nutshell

Thanks to the **RGoogleAnalytics** package (Pearmain et al., 2014), R possesses a *Google Analytics API* binding. Therefore, it is possible to access and query data from *Google Analytics* directly. Additionally, the package enables access to all *Google Analytics* accounts of a user. In terms of this section, all download and user behavior statistics for the QuantNet website can be acquired conveniently through that package [9].

The **RGoogleAnalytics** package allows for integration of data from *Google Analytics* into the R environment. Six functions are included, which will be presented here. Because

[9] QuantNet first created a *Google Analytics* account on 16th of November 2013 and all statistics can only be retrieved starting then

*Google Analytics* and R are two different application environments, a connection needs to be established for further processing of any kind of data.

```
oauth_token <- Auth(client.id = "XXX", client.secret = "YYY")
save(oauth_token, file = "oauth_token")
load("oauth_token")
ValidateToken(oauth_token)
```

The `Auth` function serves that purpose and is the necessary first step. Precisely speaking, it authorizes the **RGoogleAnalytics** package to the user's *Google Analytics* account using *OAuth2.0*, an open protocol for standardized and secure *API*-authorization between applications. Two arguments are required: `client.id` resembles the user name and `client.secret` the corresponding pass-phrase. The created token can be saved to a file and, in subsequent runs, called up again without requiring the user's consent. Only if the user queries another *Google Analytics* profile with another email account, that consent is required again. However, that token has a 60 minute lifetime, after which a new token can be obtained by using the `ValidateToken` method. This method checks if the token is expired, and if this is true, a new token will be generated and the token object updated.

```
GetProfiles(oauth_token)
```

Continuing from there, the `GetProfiles` function creates a data frame of all profile IDs and profile names by using the created token as the only argument. After retrieving all profile information, *Google Analytics* query parameters need to be initialized.

```
query.params.list <- Init(start.date = NULL, end.date = NULL, dimensions = NULL,
    metrics = NULL, filters = NULL, sort = NULL, segments = NULL, max.results = NULL,
    start.index = NULL, table.id = NULL)
```

This can be done by the `Init` function. It combines all query parameters into a list. Parameters like `start.date` and `end.date` define the timeframe for the requested *Google Analytics* data. Up to 7 dimensions can be set by the `dimensions` argument and up to 10 metrics by the `metrics` argument. In both cases this can be done as a single `string` or as a vector of `strings`. Besides setting the scope, several arguments can be used to preprocess the retrieved data. With `sort`, the sorting order of the returned data can be set. Additionally, a filter string for the *Google Analytics* request can be used to narrow the data for processing. The `segments` argument serves the purpose to slice and dice the data to define segments. Finally, the `start.index` and `max.results` arguments set the first row and the following number of rows, which will be included in the query response. The `table.id` depicts the *Analytics View ID*, for which the query will retrieve the data. All query parameters are `NULL` by default. Listing 5 demonstrates the use of the parameters. After initializing the query list, we pass it on to the `QueryBuilder` function.

```
ga.query <- QueryBuilder(query.params.list)
```

The function `QueryBuilder` initializes an object with all query parameters and validates them. The created object `qa.query` in combination with the created token are passed on to the `GetReportData` function.

```
GetReportData(ga.query, oauth_token, split_daywise = FALSE, paginate_query = FALSE)
```

Additional optional arguments like `split_daywise` and `paginate_query` are available: the first one splits the query into day-wise partitions by date range, the second one numbers chunks of results by requesting a maximum number of allowed rows at a time. Finally, the `GetReportData` function retrieves the requested data from the *Core Reporting API*.

## 5.3 Metrics, Dimensions, Event Tracking in Google Analytics

In general, a metric is a quantitative measurement of statistics describing events or trends on a website. A key performance indicator (KPI) is a metric that helps to understand how you are doing against your objectives (Kaushik, 2010).

In *Google Analytics*, *metric* is a number, which is used to measure one of the characteristics of a dimension. A *dimension* is the attribute of visitors to a given website. Taken together, a dimension provides context to a metric. Though both dimensions and metrics are the characteristics of the website visitors, they are different in the way they are configured, collected, processed, reported and queried in *Google Analytics*.

*Event Tracking* captures data differently from the standard tag-based Page View data. The event data is stored differently and creates new metrics that capture the unique experience of rich media and user actions triggered by click events or by keyboard entries. The *Event Tracking - Dimensions and Metrics Reference* describes all event tracking dimensions and metrics available in the *Real Time Reporting API*.

## 5.4 Most downloaded Quantlets: a code example

| Quantlet | Downloads |
|---|---|
| autocorr.m (autocorrelation plots) | 2450 |
| MVACARTBan1 (US bankruptcy analysis) | 677 |
| SFSmeanExcessFun (generalized Pareto distribution) | 393 |
| SFEVolSurfPlot (implied volatility surface) | 371 |
| MVAandcur (Andrew's curves) | 363 |
| SMSboxcar (Boxplot car mileage) | 339 |
| blsprice (Black-Scholes price function) | 331 |
| IBTblackscholes (call & put options - Black Scholes) | 327 |

```
query.list.Qlet <- Init( start.date = "2013-11-16", end.date = "today",
       dimensions = "ga:eventLabel", metrics = "ga:totalEvents",
       filters = "ga:eventCategory==QNetShow", sort = "-ga:totalEvents",
       max.results = 2000, table.id = "ga:78690351")
ga.query <- QueryBuilder(query.list.Qlet)
ga.df <- GetReportData(ga.query, oauth_token)

samplesize = 8
colnames(ga.df) = c("Quantlet", "Downloads")
# output as data frame, top downloaded Quantlets as defined by samplesize
ga.df[1:samplesize,]
```

Listing 5: **RGoogleAnalytics** code for extracting the most downloaded Quantlets

The presented script in Listing 5 retrieves the most downloaded QLs. It extracts recent download statistics from *Google Analytics* starting November 2013 for each QL. Furthermore, short explanations (based on the desciptions in the meta information) of the specific QLs are added and the final results are presented as an R data frame. The API query specifies the parameters `dimensions`, `metrics` and `filters` conditioning the desired *Event Tracking* criteria. Every time the user clicks on a Quantlet page, this interaction is tracked by *Event Tracking* and is available for further analysis. The R code for the full reproducibility with additional postprocessing (short explanations) and LaTeX-table output is available as TopDownloads.

## 5.5 Most Quantlet downloads by country: a code example

| Country | Downloads |
|---|---|
| Germany | 37042 |
| United States | 9735 |
| China | 9266 |
| Bulgaria | 2502 |
| Russia | 2356 |
| United Kingdom | 2265 |
| India | 1653 |
| Italy | 1643 |
| Japan | 1609 |
| France | 1335 |

```
dimensions = "ga:country" # in the 'Init' function
samplesize = 10
colnames(ga.df) = c("Country", "Downloads")
```

Listing 6: **RGoogleAnalytics** code for extracting the most Quantlet downloads by country

Listing 6 shows only the required changes relative to the code in Listing 5, see DownloadsByCountry for the full code. Within the `Init` function, only the parameter `dimensions` needs to be adjusted. While the metric and retrieved raw data remain the same, the final information is aggregated by the new dimension "country". The remaining two changes concern the format of the output data frame. It should be noted that the web analytics results from Listing 6 differ from those in Figure 3. The first results reflect the download statistics triggered by user events (mouse click), the latter reflect the total user sessions (as defined in *Audience Overview*) in the same time period. This is exactly the difference between the *Event Tracking* and *Page View* approach, see Section 5.3. During the same session a user can perform several user events like mouse clicks.

## 5.6 Most frequent search queries: a code example

The code example in Listing 7 demonstrates the main part of the procedure how to retrieve the most frequent search queries entered into the search field of the QuantNet visualization. See the complete QTopSearch for all details. The R code uses the same

`dimensions` and `metrics` parameters as in the Listing 5. The main difference are other `filters` (`eventCategory = QNet2Visu` and `eventAction = search`) and `table.id` settings. The latter parameter specifies from which *Google Analytics view* (profile) to retrieve data. The QuantNetXploRer in Figure 5 is the "source" for events which are observed by the profile with the `table.id` = 134092861.

```
query.list.search <- Init( start.date = s_date, end.date = "today",
        dimensions = "ga:eventLabel", metrics = "ga:totalEvents",
        filters = "ga:eventCategory==QNet2Visu;ga:eventAction==search",
        sort = "-ga:totalEvents", max.results = 1000, table.id = "ga:134092861")
ga.query <- QueryBuilder(query.list.search)
ga.df <- GetReportData(ga.query, oauth_token)

samplesize = 20
colnames(ga.df) = c("SearchQuery", "frequency")
ga.df[1:samplesize,]
```

Listing 7: **RGoogleAnalytics** code for extracting the most frequent search queries

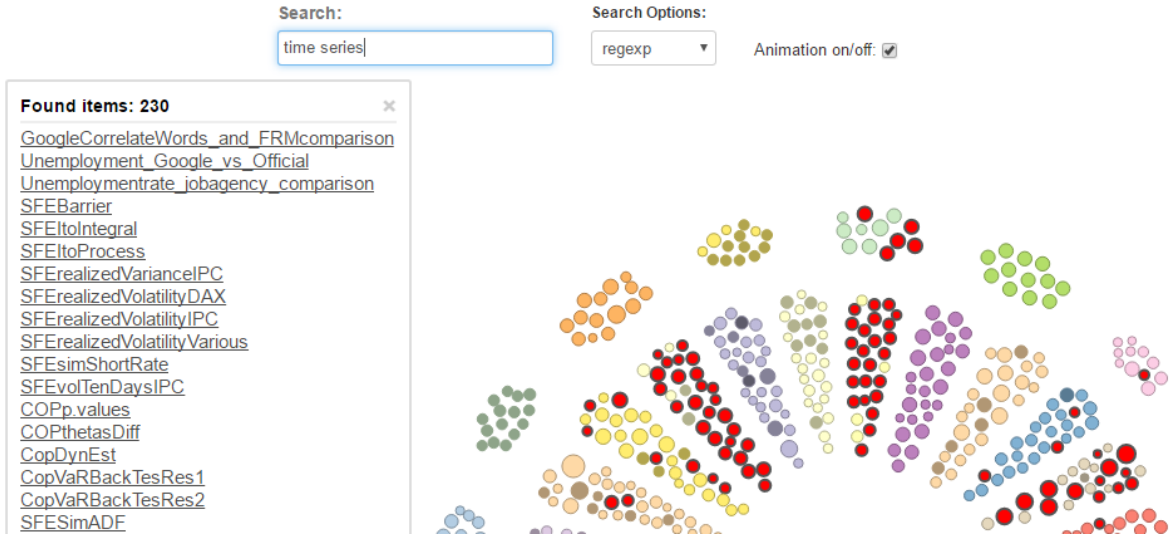


Figure 5: Search field in the QuantNetXploRer: after every keystroke the Quantlets relevant to the search query are displayed both in textual form as in graphical form; additionally the search queries are tracked via Google Analytics

As we will see in the next section, the most frequent search queries obtained from *Google Event Tracking metrics* can be used as queries in the *test collections* in order to evaluate and calibrate the information retrieval (IR) performance in different TM models. In this manner, we have a kind of "Google Analytics driven" dynamic test queries for IR evaluation allowing to concentrate on the most frequent and hence most popular search queries, which could help to reduce the efforts usually encountered in IR relevance assessments.

# 6 IR in a nutshell

An examination of the opening pages of a number of information retrieval (IR) books reveals that each author defines the topic of IR in different ways. Some say that IR is simply a field concerned with organizing information (Salton, 1968), and others emphasize the range of different materials that need to be searched (Witten et al., 1999). While others stress the contrast between the strong structure and typing of a database system with the lack of structure in the objects typically searched in IR (Rijsbergen, 1979). Across all of these definitions, there is one thing in common: IR systems have to deal with incomplete or underspecified information in the form of the queries issued by users. The IR systems must be able to handle the users' underspecified query.

The typical interaction between a user and an IR system has the user submitting a query to the system, which returns a ranked list of objects that hopefully have some degree of relevance to the user's request with the most relevant at the top of the list. Sections 3 and 6.5 provide some examples. The success of such an interaction is affected by many factors, the range of which has long been considered. Sanderson (2010) discusses the following five:

1. The ability of the system to present all relevant documents
2. The ability of the system to withhold non-relevant documents
3. The time interval between the demand being made and the answer being given
4. The physical form of the output (i.e., presentation)
5. The effort, intellectual or physical, demanded of the user

Sanderson (2010) points out that many other factors can be considered, some of them are: 1) the ability of the user at specifying their need, 2) the interplay of the components of which the search algorithm is composed, 3) the type of user information need.

## 6.1 Test Collections

A strong focus of IR research has been on measuring the effectiveness of an IR system: determining the relevance of items, retrieved by a search engine, relative to a user's information need. The vast majority of published IR research assessed effectiveness using a resource known as a *test collection*, applying thereby evaluation measures. There are many conferences and meetings devoted purely to *test collections*, including three international conferences, TREC, CLEF, and NTCIR, which together took place more than 30 times since the early 1990s. This research focus is a part of a longer tradition which was motivated by the creation and sharing of testing environments in the previous three decades, which itself was inspired by innovative work conducted in the 1950s.

The classic components of a *test collection* are as follows: 1) a *collection of documents*, 2) a *set of topics* (also referred to as queries), 3) a *set of relevance judgments* composed of a list of topic/document pairs, detailing the relevance of documents to topics.

The genesis of IR evaluation is generally seen as starting with the work of Cleverdon and his Cranfield collections, built in the early 1960s. However, he and others were working

on retrieval evaluation for most of the 1950s (Cleverdon, 1959). In his first collection Cleverdon tested four competing indexing approaches on a set of 18.000 papers. The papers were manually indexed using each of the four classification methodologies. Once the indexes were built, the papers were searched with 1.200 "search questions". The collection became known as Cranfield I. Cleverdon concluded that the relatively large size of Cranfield I was not important in ensuring reliable measurements. Therefore, the new collection was composed of 1.400 "documents" (titles, author names and abstracts) derived from the references listed in around 200 recent research papers. This work resulted in the Cranfield II collection comprising 1.400 documents, 221 topics, and a set of complete variable level relevance judgments. Alongside the work of Cleverdon, Salton initiated the creation of a series of test collections, collectively known as the SMART collections (Lesk and Salton, 1968). A good review of the IR research and its history is provided in Sanderson (2010).

## 6.2 Effectiveness measures: Precision, Recall

| | Relevant | Non-relevant | |
|---|---|---|---|
| Retrieved | $a$ | $b$ | $a+b$ |
| Not retrieved | $c$ | $d$ | $c+d$ |
| | $a+c$ | $b+d$ | $a+b+c+d$ |

Table 5: Contingency table of all possible quantities in IR

Cleverdon et al. (1966) produced a *contingency table* of all possible quantities that could be calculated to judge an information retrieval system. Table 5 is reproduced including the original labels. Another common notation for the table's cells in Table 5 is: *true positives* (a), *false positives* (b), *false negatives* (c) and *true negatives* (d), see Manning et al. (2008). Various measures can be created out of combinations of the table's cells. The three that are probably the best known are:

$$\text{Precision} = \frac{a}{a+b}\ , \qquad \text{Recall} = \frac{a}{a+c}\ , \qquad \text{Fallout} = \frac{b}{b+d}\ . \tag{1}$$

Where *precision* measures the fraction of retrieved documents that are relevant, *recall* measures the fraction of relevant documents retrieved and *fallout* measures the fraction of non-relevant documents retrieved. Most IR systems experience a dilemma concerning them. To improve a system's precision, the system needs strong measures for deciding whether a document is relevant to a query. This will help minimize the *false hits*/false positives (quantity $b$ in Table 5), but it will also affect the number of relevant documents that are retrieved. These strong measures can prevent some important relevant documents from being included within the set of documents that satisfy the query, thereby lowering the recall (Herbert et al., 2004). Further, because the number of relevant documents in a set of documents is fixed for any query (it is the quantity $a+c$ in Table 5) we can use the true positives (quantity $a$) as proxies for the "relative recall". In practice the determination of the total number of relevant documents relative to a query is difficult and time-consuming due to the size of the document corpus. A higher true positive value implies a higher recall value, up to the scalar multiple of $\frac{1}{a+c}$.

## 6.3 Google Analytics driven QuantNet Test Collection

As data set for the following IR performance analysis, the current YAML meta information was parsed from QuantNet by means of the TM pipeline, see Figure 1. Hence, our *collection of documents* contains 1140 YAML documents. Our *set of topics* was formed from the most frequent search queries delivered by the Google Event Tracking metrics, see Section 5.6. The *set of relevance judgments* is determined on demand, when additional precision and recall benchmarks are required. All results and tables were calculated by use of the **TManalyzer** package, which is the TM layer of the "GitHub API based QuantNet Mining infrastructure in R".

Throughout the rest of our article we will use the definitions and notations from Section **Vector space representations** in Borke and Härdle (2017). The most important quantities are: $Q = \{d_1, \ldots, d_n\}$ as a set of documents/Quantlets; $T = \{t_1, \ldots, t_m\}$ as a dictionary (set of all terms); $tf(d, t)$ as the absolute frequency of term $t \in T$ in $d \in Q$; *tf-idf* as the term frequency - inverse document frequency; $D$ as a "term by document matrix" TDM. The TM models BVSM, GVSM(TT) and LSA were considered. From LSA two configurations were examined: LSA (50% of the weight of all singular values maintained) and LSA50 (with the dimension parameter $k = 50$).

## 6.4 IR system designs in 3 models

All IR results and TDM representations in this section were calculated by means of the **TManalyzer** package as displayed in Listing 13.

### 6.4.1 Single term queries

Via **RGoogleAnalytics** (see Listing 7) the following most frequent single term search queries were obtained: "covar", "random", "quantile", "histogram", "multivariate", see Table 6. They serve as the first *set of topics* in the IR simulation in Listing 13. A document is retrieved if its similarity value relative to the query is bigger than the given IR threshold. Taking different weighting schemes (argument `tf_weight`) and varying threshold levels (argument `sim_threshold`), we obtain the IR results as summarized in Table 7.

| | q1 | q2 | q3 | q4 | q5 |
|---|---|---|---|---|---|
| covar | 1 | 0 | 0 | 0 | 0 |
| histogram | 0 | 0 | 0 | 1 | 0 |
| multivari | 0 | 0 | 0 | 0 | 1 |
| quantil | 0 | 0 | 1 | 0 | 0 |
| random | 0 | 1 | 0 | 0 | 0 |

Table 6: TDM of the single term queries in the post processed raw TF-form

The results in Table 7 can be summarized as follows. The LSA50 model retrieves the most documents for all `tf_weight` × `sim_threshold` combinations. BVSM returns the

least number of matches. Under the weighting scheme $tf$ no clear dominance between GVSM(TT) and LSA can be determined. Under the weighting scheme $tf$-*idf* GVSM(TT) outperforms LSA in many cases. Additionally, the weighting scheme $tf$-*idf* retrieves more matches averaged over all TM models × `sim_threshold` combinations.

| | B | TT | LSA | L50 | B | TT | LSA | L50 | B | TT | LSA | L50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| weighting scheme | tf normalized | | | | | | | | | | | |
| covar | 0 | 0 | 0 | 7 | 0 | 0 | 3 | 9 | 0 | 0 | 6 | 9 |
| random | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 7 | 0 | 10 | 3 | 18 |
| quantile | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 2 | 6 |
| histogram | 0 | 0 | 0 | 3 | 0 | 0 | 2 | 6 | 2 | 2 | 4 | 14 |
| multivariate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 16 |
| weighting scheme | tf-idf normalized | | | | | | | | | | | |
| covar | 0 | 0 | 2 | 7 | 0 | 4 | 5 | 9 | 0 | 6 | 6 | 11 |
| random | 0 | 0 | 0 | 11 | 0 | 3 | 2 | 17 | 0 | 13 | 9 | 24 |
| quantile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 2 | 17 |
| histogram | 0 | 2 | 1 | 13 | 1 | 10 | 7 | 19 | 3 | 16 | 13 | 26 |
| multivariate | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 12 | 0 | 12 | 0 | 21 |

Table 7: Number of QLs retrieved in each of 3 TM models; measure: cosine similarity; similarity threshold for IR: 0.8, 0.7, 0.6 (from left to right)

### 6.4.2 Compound term queries

Via **RGoogleAnalytics** (see Listing 7) the following most frequent compound term search queries were obtained: "random number", "multivariate statistics", "black scholes", see Table 8. The second *set of topics* for the IR simulation in Listing 13 yields the results presented in Table 9.

| | q1 | q2 | q3 |
|---|---|---|---|
| black | 0 | 0 | 1 |
| multivari | 0 | 1 | 0 |
| number | 1 | 0 | 0 |
| random | 1 | 0 | 0 |
| schole | 0 | 0 | 1 |
| statist | 0 | 1 | 0 |

Table 8: TDM of the compound term queries in the post processed raw TF-form

The results in Table 9 show a similar situation as in the case of "single term queries". But GVSM(TT) is clearly better than LSA. LSA50 is best and BVSM is still worst (concerning the number of hits). One remarkable observation in all Tables (7 and 9) is that LSA50 clearly outnumbers the other models at the highest IR threshold = 0.8. The other models have none or very few hits.

| | B | TT | LSA | L50 | B | TT | LSA | L50 | B | TT | LSA | L50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| weighting scheme | tf normalized | | | | | | | | | | | |
| random n. | 0 | 1 | 0 | 7 | 0 | 8 | 2 | 7 | 1 | 8 | 6 | 15 |
| multivariate s. | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | 11 | 0 | 11 |
| black s. | 0 | 1 | 0 | 20 | 0 | 44 | 0 | 46 | 0 | 58 | 1 | 55 |
| weighting scheme | tf-idf normalized | | | | | | | | | | | |
| random n. | 0 | 1 | 0 | 9 | 0 | 6 | 1 | 17 | 1 | 14 | 5 | 23 |
| multivariate s. | 0 | 0 | 0 | 6 | 0 | 4 | 0 | 15 | 0 | 14 | 0 | 22 |
| black s. | 0 | 3 | 0 | 43 | 0 | 43 | 1 | 51 | 0 | 52 | 1 | 59 |

Table 9: Number of Qs retrieved in each of 3 TM models; measure: cosine similarity; similarity threshold for IR: 0.8, 0.7, 0.6 (from left to right)

## 6.5 IR Performance: recall and precision in 3 models

In order to assess the IR effectiveness and validity of the previous results, we have to examine the relevance of the retrieved documents for each query individually. This is demonstrated by two examples.

```
# Single term queries
query = c("covar", "random", "quantile", "histogram", "multivariate")

query.tm.folded = query.tm.fold_in(query, tm_list, tf_weight = "ntc")
q_tdm_sim.tm_res = q_tdm_sim.tm.list(query.tm.folded)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8)
q_ir_list$query_tm_text["histogram"]
# returns the retrieved docs for every TM model, see below for the full output
"BVSM: no hits\\GVSM(TT): SPMHistoConstruct (0.9), SPMhistobias2 (0.82)\\LSA:
    SPMHistoConstruct (0.9)\\LSA50: SPMhistobias2 (0.97), SPMHistoConstruct (0.96)..."

q_ir_list$query_tm_list[["histogram"]]
 [1] "SPMHistoConstruct" "SPMhistobias2" "BCS_hist1" "BCS_HistBinSizes" "BCS_hist2"
 [6] "SPMbuffagrid" "SPMbuffahisto" "SPMhistogram" "SPMashstock" "SPMstockreturnhisto"
 [11] "SPMhiststock" "SPMsimulatedexponential" "SPMbuffadata"
```

Listing 8: IR results inspection via **TManalyzer**

The extended IR results for the search query "histogram" (tf-idf, IR threshold 0.8, numbers in brackets show the similarity values) are produced by the function `query.similar.doc.inspect` and stored in the variable `q_ir_list$query_tm_text`, see Listing 8:

**BVSM**: no hits; **GVSM(TT)**: SPMHistoConstruct (0.9), SPMhistobias2 (0.82);
**LSA**: SPMHistoConstruct (0.9); **LSA50**: SPMhistobias2 (0.97), SPMHistoConstruct (0.96), BCS_hist1 (0.95), BCS_HistBinSizes (0.92), BCS_hist2 (0.91), ~~SPMbuffagrid~~ (0.9), SPMbuffahisto (0.89), SPMhistogram (0.89), SPMashstock (0.89), SPMstockreturnhisto (0.87), SPMhiststock (0.87), SPMsimulatedexponential (0.85), ~~SPMbuffadata~~ (0.82)

Manual inspection of the retrieved QLs for the query "histogram" shows that there are two *false hits* (SPMbuffagrid, SPMbuffadata) in the LSA50 model. The other TM models have maximum precision (i.e. $Precision = 1$), whereas the precision of LSA50 is $\frac{11}{13}$. The (relative) recall performance is: LSA50 $>_{recall}$ GVSM(TT) $>_{recall}$ LSA $>_{recall}$ BVSM, see

Section 6.2 for IR effectiveness quantities and notations.

The IR results for the search query "random number" (tf-idf, IR threshold 0.7) were produced in an analogous way as in Listing 8:
**BVSM**: no hits; **GVSM(TT)**: SFEfibonacci (0.86), SFErandu (0.79), SFErangen2 (0.76), SFErangen1 (0.75), random_walk (0.74), SFEBMuller (0.71); **LSA**: SFEfibonacci (0.78);
**LSA50**: BCS_LFG (0.97), SFErandu (0.95), SFEfibonacci (0.95), SFErangen2 (0.95), SFErangen1 (0.94), BCS_RANDU (0.9), BCS_ARM (0.86), BCS_Shapes (0.85), random_walk (0.81), SFEBMuller (0.8), SFEevt3 (0.78), randomwalk_ar1 (0.77), simulationplot (0.76), SFEtrinomp (0.72), MSMasprob (0.72), SFEevt2 (0.72), BCS_claytonMC (0.7)

Manual inspection of the retrieved QLs for the query "random number" shows that all hits are relevant. Hence, we can conclude that all TM models provide maximum precision, i.e. all retrieved documents are relevant, and:
LSA50 $>_{recall}$ GVSM(TT) $>_{recall}$ LSA $>_{recall}$ BVSM.

The manual inspection of all single term queries' results has revealed that there are two *false hits* (tf-idf, IR threshold 0.8). Incorporating this information into the function `query.similar.doc.inspect` (see Listing 9) allows to build three matrices as shown in Table 10: $M_{retrieved}$, $M_{true\_positives}$, $M_{precision}$. We have thus for each query × TM model combination the number of retrieved documents, the number of retrieved and relevant documents (true positives) and the precision value. We can conclude that the number of true positives for all single term queries is maximal in the LSA50 model. Except the combination "histogram"/LSA50 all other cases (query × TM model combinations) show maximum precision.

```
false_hits = list("histogram" = c("SPMbuffagrid", "SPMbuffadata"))

q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8,
                                      false_hits = false_hits)

m_retr = q_ir_list$retrieved_m
m_true_positives = q_ir_list$relevant_m
m_precision = q_ir_list$relevant_m / q_ir_list$retrieved_m
m_precision[is.nan(m_precision)] = 0

colnames(m_retr) = colnames(m_true_positives) = colnames(m_precision) =
                   c("B", "TT", "LSA", "L50")
( m_IR = cbind(m_retr, m_true_positives, round(m_precision, 2)) )
```

Listing 9: IR effectiveness inspection via **TManalyzer**

| | B | TT | LSA | L50 | B | TT | LSA | L50 | B | TT | LSA | L50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| covar | 0 | 0 | 2 | 7 | 0 | 0 | 2 | 7 | 0 | 0 | 1 | 1.00 |
| random | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 1.00 |
| quantile | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00 |
| histogram | 0 | 2 | 1 | 13 | 0 | 2 | 1 | 11 | 0 | 1 | 1 | 0.85 |
| multivariate | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 1.00 |

Table 10: IR performance for single term queries, tf-idf, IR threshold 0.8: $M_{retrieved}$, $M_{true\_positives}$, $M_{precision}$ (from left to right)

# 7 Cluster Validation

Clustering plays a major role in dealing with high-dimensional data. Being first used for simple classifications, clustering ended up as an integral part of different disciplines like archeology, linguistics, bioinformatics, genetics and others (Everitt et al., 2011). Nowadays a vast amount of various numerical methods are introduced in order to make the retrieval of information from the clustering partition easier and more efficient and also to assess how efficient it is.

Our main goal here is to try different clustering and validation methods within the R software environment and to determine the optimal combination of data configuration, vector space model, clustering method and clustering criteria settings, thereby evaluating the resulting partition and eventually finding a reasonable number of clusters. The so-called "5-level Validation Pipeline" is described in Section 7.1.

Depending on the matrix $P$ (an appropriate linear transformation), we will consider three TM models, examined in Cristianini et al. (2002) and evaluated in the $M^3$-benchmark as described in Borke and Härdle (2017): BVSM, GVSM(TT) and LSA. Furthermore, we will compare three different clustering methods: hierarchical clustering, $k$-means and $k$-medoids (or pam - partitioning around medoids). All of them are rather well-known and often used, see for more details Everitt et al. (2011).

For validation of clustering methods different measures (also called clustering criteria, clustering validity indices or quality indices) were introduced. We will consider those of them implemented in the R packages **clusterCrit** (Desgraupes, 2016) and **NbClust** (Charrad et al., 2014). The optimal number of clusters can be derived by maximizing/minimizing of the index value or the difference between two successive slopes. The last means that on a plot with index values $Q$ against the number of selected clusters $K \in \{K_m, \ldots, K_M\}$, the best value for $K$ corresponds to an elbow. Suppose, for example, we need to maximize the difference between two successive slopes. Let us denote $V_i = Q_{i+1} - Q_i$, then K is determined by:

$$K = \arg \max_{K_m < i < K_M} (V_i - V_{i-1}).$$

Some of the 27 internal quality indices offered by the package **clusterCrit** are of the exceptional interest, allowing rather clear interpretation. The following list contains the names (as used in Desgraupes (2013)) of 12 measures that we have selected for our benchmark: Ball-Hall, C-Index, Calinski-Harabasz, Davies-Bouldin, Dunn, McClain-Rao, Ratkowsky-Lance, Ray-Turi, Silhouette, Trace-W, Wemmert-Gancarski and Xie-Beni.

All these clustering validity indices combine information about intracluster compactness and intercluster isolation, as well as other factors, such as geometric or statistical properties of the data, the number of data objects and dissimilarity or similarity measurements. More about the theory, index formulas and additional information can be found in Brock et al. (2008), Desgraupes (2013) and Charrad et al. (2014).

## 7.1 Validation Pipeline

The experimental design for cluster validation presented here is a direct evolution of the $M^3_{d_1,d_2,d_3,max}$ design as introduced in Borke and Härdle (2017). Additionally, we embed another new dimension into the concept, namely different configurations of meta information, thus expanding $M^3_{d_1,d_2,d_3,max}$ to $M^4_{d_1,d_2,d_3,d_4,max}$. The new $M$ stands for the new dimension "meta information".

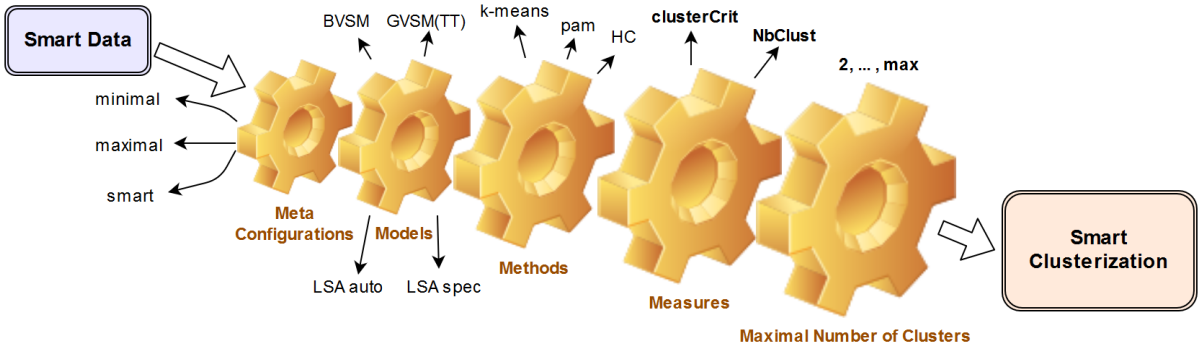


Figure 6: Validation Pipeline $M^4_{d_1,d_2,d_3,d_4,max}$

Let us call this performance validation approach the **Validation Pipeline** (*Vali-PP*). The main goal of this validation benchmark is to calibrate such a combination of all 5 dimensions (*Vali-PP*-configurations), that provides the best clustering result, when applied to YAML data. An intuitive interpretation of this idea could be a pipe with 5 gear wheels (each of them representing one optimization component/parameter), that takes preprocessed (smart) data as an input and returns an optimal smart clusterization as an output. A schematic illustration of the described process is displayed in Figure 6.

The main purpose of this analogy is to determine the best angle of rotation for each gear wheel, so that together their combination lets smart data pass through the pipe in the most effective way. The term "effectiveness" means here that we should try to find a compromise between the information gain from our data and the dimension reduction of data at the same time, omitting unnecessary information and hence reducing the storage and computational costs.

### 7.1.1 Vali-PP in a nutshell

Summarized, the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$ deals with the following 5 dimensions:

- $d_1$: 3 sample configurations of **meta information**: minimal, maximal, smart
- $d_2$: 3 vector space **models** / 4 TM configurations: BVSM, GVSM(TT), LSA (automatic choice of the dimension and own choice of the dimension, e.g. LSA25), encoded in "standard TM colors": BVSM, GVSM(TT), LSA, LSA25
- $d_3$: 3 clustering **methods**: $k$-means, $k$-medoids, Hierarchical Clustering (HC) (comprising average, ward.D and ward.D2 agglomeration methods)

- $d_4$: 12 **measures** (quality indices) for validating the clustering quality (from R packages **clusterCrit** and **NbClust**)
- *max*: **maximal** number of clusters: from 2 to 250 (YAML-500), from 2 to 100 (YAML-1140)

Hence, the *Vali-PP* process encompasses $3 \times 4 \times 3 \times 12 \times 249 = 107.568$ different combinations (*Vali-PP*-configurations) in the maximum case, each of them needing to be evaluated.

### 7.1.2 The new dimension

By "configuration of meta information" we mean a weighted combination of data fields which are extracted from the YAML meta information and which are used in the subsequent validation process. In the course of our study we propose three sample configurations, two extreme cases and one of our own choice. The "configurations of meta information" can be easily extracted by means of the packages **yamldebugger** and **TManalyzer**. Listing 10 demonstrates the use of the both packages. The function `yaml.list.extract` (from **TManalyzer**) relies on the function `yaml.getQField` (from **yamldebugger**).

```
library(yamldebugger)
library(TManalyzer)
(obj.names = load("yaml_list_full_20161122.RData", .GlobalEnv))  # 1140 Docs

help(yaml.getQField)
yaml.getQField(yaml_list[[1]], "d")

# Meta Configuration I
 t_vec = yaml.list.extract(yaml_list, weight = c(q=1, d=1, k=1, p=1))
# Meta Configuration II
 t_vec = yaml.list.extract(yaml_list)
# Meta Configuration III
 t_vec = yaml.list.extract(yaml_list, weight = c(d=6, k=10, sa=3, a=4, df=5, e=4))

str(t_vec)
> List of 2
> $ t_vec : chr [1:1140] " MSMLLN Plots the points showing law of large numbers." ...
> $ q_names: chr [1:1140] "MSMLLN" "MSM_LIL" "MSM_VaRandES" "MSMasprob" ...
```

Listing 10: Extraction of Meta Configurations via **yamldebugger** and **TManalyzer**

1. **Configuration I**: Text documents are represented by descriptions and keywords data fields only, this case is thus minimum that could be reasonable for text mining and the most relevant as well.
2. **Configuration II**: Text documents are represented by all (not technical) data fields of the meta info, and this case is obviously the maximum one, including though a considerable amount of potentially uninformative and unreliable content.
3. **Configuration III**: Text documents are represented by weighted combinations of the most substantial fields (concerning the amount of words). Weights chosen were (6, 10, 3, 4, 5, 4) for the fields: "description", "keywords", "see also", "author", "datafile" and "example" correspondingly. The logic behind such a choice is that

"description" and especially "keywords" reflect the most profound essence of the text and must thus make the greatest contribution. The next most important category is "datafile", since several QLs related to the same observation data set should be more or less close to each other. Slightly less relevant are "author" and "example", because the author can also submit QLs from a completely different area and "example" often just duplicates the "description" content. Then at the end comes "see also" with only names of the similar documents in it.

## 7.2 Experimental Procedure

For the clustering validation part of the *Vali-PP*, two R packages were used: **clusterCrit** and **NbClust**. By means of the first library we calculated all 12 quality indices (measures) as described before. From the second package those eight measures were taken, which intersect with the measures from **clusterCrit**, namely: Ball-Hall, C-Index, Calinski-Harabasz, Davies-Bouldin, Dunn, McClain-Rao, Ratkowsky-Lance and Silhouette. We had several reasons to include another library into our study: 1) to compare results for the same validation method from different packages; 2) to recalculate results for Silhouette index which were partially corrupted (and obviously useless) in the **clusterCrit** implementation; 3) to test whether those measures in **clusterCrit** that did not provide clear interpretation results could be easier interpreted using the other package. Within the package **clusterCrit** we compared the methods $k$-means, $k$-medoids and HC (*average* agglomeration), whereas from **NbClust** HC (*ward.D* and *ward.D2* agglomeration) and $k$-means were selected.

Two data sets were examined. The first one, YAML-500, was extracted from the YAML meta information of 500 QLs on December 26, 2015, see Section 7.3. The second one, namely YAML-1140, was collected from 1140 QLs on November 22, 2016 and used in Section 7.4. The choice of 250 as maximum number of clusters for YAML-500 is justified as in Section "**3 Models, 3 Methods, 3 Measures**" in Borke and Härdle (2017). For YAML-1140 the cluster range within the interval $\{2, \ldots, 100\}$ was selected. Since the comprehensive $M^4_{d_1,d_2,d_3,d_4,250}$ analysis of the YAML-500 data has revealed the general properties of the *Vali-PP*-configurations, it was sufficient for the second stage to concentrate on the practically relevant cluster sizes, see last paragraph in Section 7.4.

## 7.3 Validation Pipeline Results

Table 11 shows the optimal TM model, clustering method and meta configuration ($d_1 \times d_2 \times d_3$) for each index, see Section 7.1.1 for the notation definitions. The last two dimensions/wheels of the *Vali-PP* (the appropriate index $d_4$ and the optimal number of clusters in the range $2, \ldots, max$) are to be considered in each particular case separately. However, some index results do not allow an unambiguous decision: Davies-Bouldin, Ray-Turi, Xie-Beni and Dunn. In these cases, necessary remarks are provided: **EI** - easy to interpret, **HI** - hard to interpret. Concerning the main results, the following (partly abbreviated) notations were used: TT for GVSM(TT), LSA25 for LSA space reduced to 25 dimensions, HC for *average* agglomeration, Ward D for HC (*ward.D* agglomeration) and Ward D2 for HC (*ward.D2* agglomeration).

| Index | Best model | Best clustering method | Best conf. |
|---|---|---|---|
| **clusterCrit** | | | |
| Ball-Hall (EI) | TT[10]/ LSA25[11] | HC | II |
| C-Index (EI) | TT/LSA25 | HC | III |
| Calinski-Harabasz (EI) | LSA25 | $k$-medoids[12] | II |
| McClain-Rao (EI) | LSA25 | $k$-means/$k$-medoids[13] | III |
| Ratkowsky-Lance (EI) | TT/LSA25 | all | I/II |
| Trace-W (EI) | LSA/LSA25 | HC | II/III |
| Wemmert-Gancarski (EI) | LSA25 | $k$-mean/$k$-medoids[14] | III |
| Davies-Bouldin (HI) | LSA25 | $k$-medoids[15] | II |
| Ray-Turi(HI) | LSA25 | HC | III |
| Xie-Beni (HI) | BVSM[16] | HC | II |
| Dunn (HI) | BVSM[17] | HC | II |
| **NbClust** | | | |
| Ball-Hall (EI) | TT[18]/ LSA25[19] | all | I/II/III |
| C-Index (HI) | LSA25 | Ward D | II/III |
| Calinski-Harabasz (EI) | LSA25 | Ward D/ Ward D2 | II |
| McClain-Rao (EI) | LSA25 | Ward D2 | III |
| Ratkowsky-Lance (EI) | TT/LSA25 | all | I/II/III |
| Davies-Bouldin (EI) | LSA25 | $k$-means | I/II/III |
| Silhouette (EI) | LSA25 | Ward D/ Ward D2 | III |
| Dunn (HI) | BVSM[20] | Ward D2 | II |

Table 11: Main results of YAML-500 for selected **clusterCrit** and **NbClust** quality indices

The results and conclusions in Table 11 were made based on *Vali-PP* plots of YAML-500 with 4 curves (a curve for each vector space model/configuration $d_2$, in the following also referred to as *Vali-PP* graph) showing the value of a given validation index ($Y$ axis) for each number of clusters from 2 to 250 ($X$ axis). These plots with *Vali-PP* graphs were created for each combination of $d_1 \times d_3$. Altogether, we had 9 ($|d_1 \times d_3|$) *Vali-PP* plots for each index $d_4$, which resulted in $9 * 11 = 99$ *Vali-PP* plots for the package **clusterCrit** (11 indices, Silhouette index was skipped) and further $9 * 8 = 72$ *Vali-PP* plots for the package **NbClust** (8 indices). All validation results, *Vali-PP* plots and the corresponding Quantlets are available at Clustering_Validation_Pipeline.

The overall conclusion can be drawn that, when considered individually, the advantage

[10]TT model is the best wrt. optimal cluster number selection
[11]LSA25 model is the best wrt. global behavior of the curves
[12]$k$-medoids method is better, but HC is comparable from a certain cluster number size
[13]HC is also comparable in LSA25, which is the best model for this index
[14]HC is here comparable wrt. global behavior of the curves
[15]HC shows more stable behavior in all the models
[16]all models are very close in HC, in particular LSA and BVSM
[17]all models, in particular BVSM and LSA, are relatively close and are also in a quite small interval, regarding that the value range of this index is $[0, +\infty)$
[18]TT model is the best wrt. optimal cluster number selection
[19]LSA25 model is the best wrt. global behavior of the curves
[20]all models, in particular BVSM and LSA, are relatively close and are also in a quite small interval, regarding that the value range of this index is $[0, +\infty)$

of LSA or HC is sometimes not obvious, but their combination is mostly not worse than any other combination $d_2 \times d_3$ of a TM model and a clustering method. As for the meta information dimension $d_1$, no clear distinction can be made, whether configuration II or III is better. In any case, both of them outperform (or are not worse than) configuration I, which includes minimal information.

Concerning the results of the same quality indices from different packages (**NbClust** vs. **clusterCrit**), the best TM model is almost always the same: LSA25 (except from the Dunn index). The indicated optimal clustering method cannot be directly compared between the both packages, as only the $k$-means method is present as the "common element" in both cases. For the first stage of our validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$ it was sufficient to identify possible optimal combinations $d_1 \times d_2 \times d_3$ within each given package, looking at them separately from different angles: quality indices $d_4$, number of clusters $(2, \ldots, max)$ and various HC agglomeration methods.

## 7.4 Smart-Vali-PP

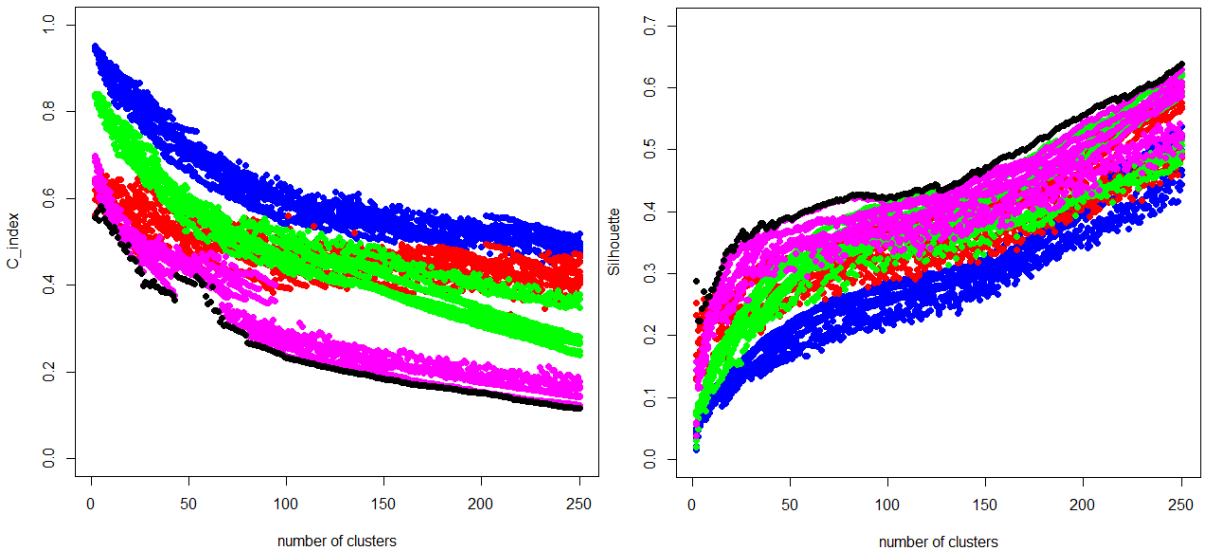


Figure 7: *Smart-Vali-plots* of YAML-500 for C-Index (left: to be minimized) and Silhouette index (right: to be maximized) from the package **NbClust**; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

The main disadvantage of the manual/visual inspection and summarization of the validation results in Section 7.3 is the tedious and sometimes rather vague analysis of the *Vali-PP* plots for any given validation index. As discussed before, there are $3 \times 3$ *Vali-PP* plots, each of them containing 4 Vali-PP graphs. Furthermore, as shown in Table 11, it is usually difficult or even impossible to characterize an optimal *Vali-PP*-configuration for a given index $d_4$ in an unambiguous way, i.e. to say there is a particular optimal $d_1 \times d_2 \times d_3$ combination (meta/model/method) for all considered cluster sizes $\{2, \ldots, max\}$. For that reason an improvised extension of the Validation Pipeline software infrastructure was developed by Lukas Borke in order to facilitate the validation inspection process. This set of functions will be referred to as "Smart-Vali-PP" in the following. *Smart-Vali-PP* is a component of the overall *Vali-PP* infrastructure, which is displayed in Figure 9. In the

near future, this code collection could be published as a separate R package.

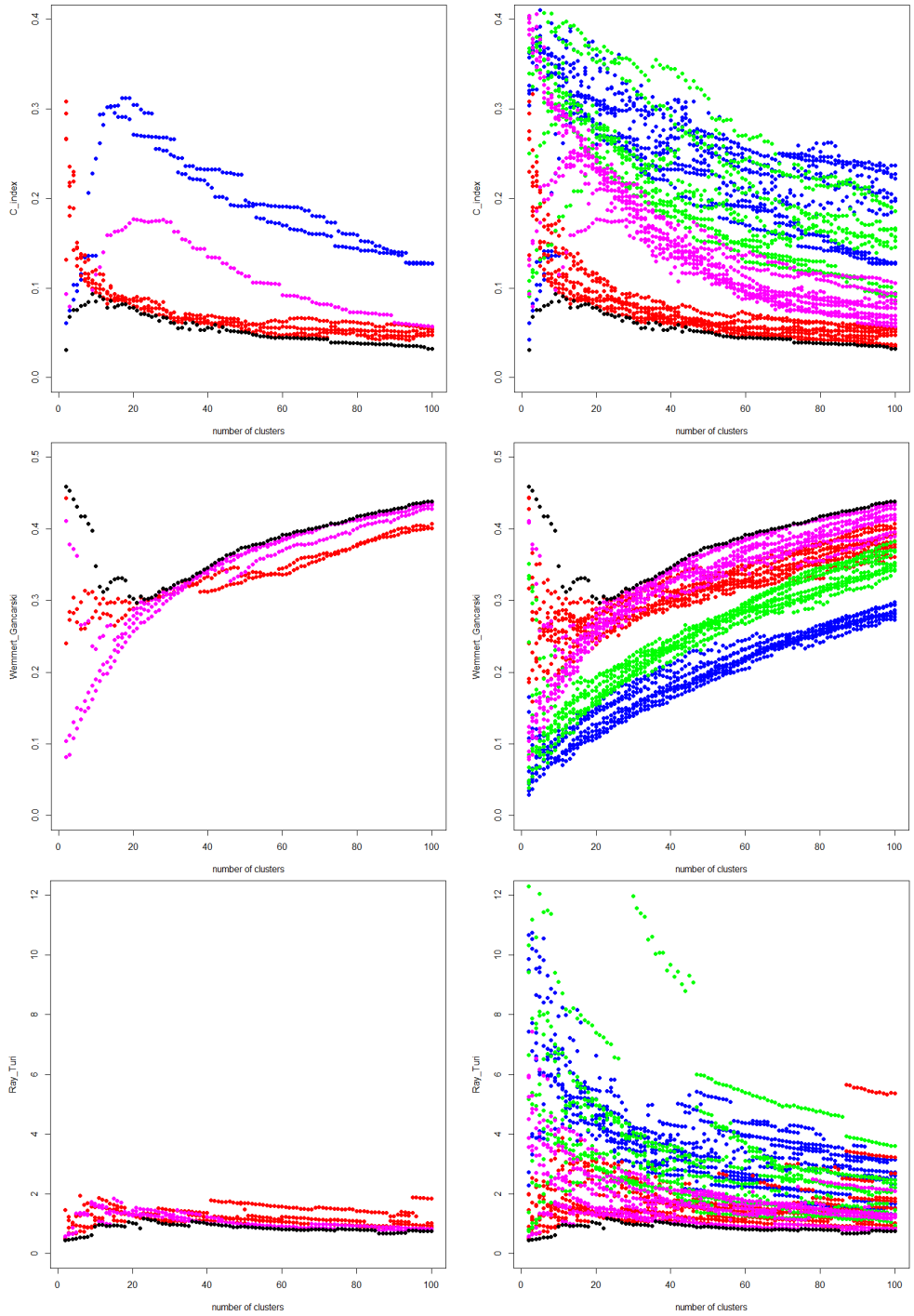


Figure 8: *Smart-Vali-plots* of YAML-1140 for the indices: C-Index, Wemmert-Gancarski, Ray-Turi from the **clusterCrit** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

The *Smart-Vali-PP* framework produces two essential outputs. One kind are the so-called "Smart-Vali-plots", which are generated using the *Vali-PP* results stored in a 5-dimensional array. For each index the aforementioned 9 plots ($d_1 \times d_3$ combination) are merged into a single plot. Together with four model configurations ($d_2$) every *Smart-Vali-plot* embeds 36 ($|d_1 \times d_2 \times d_3|$) different *Vali-PP* graphs (encoded in the standard TM colors) having the cluster size on the $X$ axis. The black curve shows the "optimal

function" (over all $d_1 \times d_2 \times d_3$ combinations) which can be achieved under the selected index. Two demonstrative *Smart-Vali-plots* are presented in Figure 7. There we can clearly see that the LSA25 model is the optimal one both under C-Index and Silhouette index.

Figure 8 shows further *Smart-Vali-plots* created for the indices C-Index, Wemmert-Gancarski and Ray-Turi. In this case, the current YAML-1140 data set was analyzed. The plots on the left side display only such *Vali-PP* graphs which intersect the "optimal function" at least at one point (one cluster size). The plots on the right side encompass all *Vali-PP* graphs, allowing the analysis of the overall shape and trends of all *Vali-PP*-configurations $d_1 \times d_2 \times d_3$. The *Smart-Vali-PP* function `plot.optimal.functions` from Listing 15 carries out the *Smart-Vali-plots* creation.

| **Conf.** $d_1$ | **Method** $d_3$ | **Model** $d_2$ | **Freq.** | **Cum. rel. prop.** | **Rel. prop.** |
|---|---|---|---|---|---|
| | | **C-Index** | | | |
| 3 | 3 | 2 | 50 | 0.50 | 0.50 |
| 3 | 2 | 2 | 15 | 0.66 | 0.15 |
| 3 | 1 | 2 | 14 | 0.80 | 0.14 |
| 1 | 1 | 2 | 10 | 0.90 | 0.10 |
| 2 | 3 | 4 | 5 | 0.95 | 0.05 |
| 1 | 2 | 2 | 3 | 0.98 | 0.03 |
| 2 | 3 | 1 | 1 | 0.99 | 0.01 |
| 3 | 3 | 1 | 1 | 1.00 | 0.01 |
| | | **Wemmert-Gancarski** | | | |
| 2 | 3 | 4 | 68 | 0.69 | 0.69 |
| 3 | 3 | 4 | 20 | 0.89 | 0.20 |
| 3 | 2 | 2 | 5 | 0.94 | 0.05 |
| 3 | 3 | 2 | 3 | 0.97 | 0.03 |
| 3 | 2 | 4 | 2 | 0.99 | 0.02 |
| 2 | 2 | 4 | 1 | 1.00 | 0.01 |
| | | **Ray-Turi** | | | |
| 3 | 3 | 4 | 38 | 0.38 | 0.38 |
| 1 | 3 | 4 | 20 | 0.59 | 0.20 |
| 3 | 3 | 2 | 18 | 0.77 | 0.18 |
| 2 | 3 | 4 | 17 | 0.94 | 0.17 |
| 2 | 3 | 2 | 4 | 0.98 | 0.04 |
| 3 | 2 | 2 | 1 | 0.99 | 0.01 |
| 1 | 3 | 2 | 1 | 1.00 | 0.01 |

Table 12: *Smart-Vali-tables* of YAML-1140 for the indices: C-Index, Wemmert-Gancarski, Ray-Turi from the **clusterCrit** package

The other kind of *Smart-Vali-PP* outputs are "Smart-Vali-tables" as shown in Table 12. For each index, the triple $d_1 \times d_2 \times d_3$ (representing a particular *Vali-PP* graph) is aggregated and sorted according to the cluster sizes where this triple coincides with the "optimal function". The additional columns "Frequency", "Cumulative relative proportion" and "Relative proportion" allow to identify and quantify the index optimality for a given *Vali-PP*-configuration $d_1 \times d_2 \times d_3$ along the dimension "cluster size". The function `optimal_share.prettify` from Listing 15 carries out the *Smart-Vali-tables* creation.

Both the *Smart-Vali-plots* and the *Smart-Vali-tables* are different representations based on the data structure `optimal_share_index`, see Listing 15. Looking for instance at the C-Index in Figure 8 and Table 12, we can easily conclude that the GVSM(TT) Vali-PP graphs are mostly in proximity to the "optimal function". Additionally, there are one LSA50 Vali-PP graph and two BVSM Vali-PP graphs which intersect the goal function at least at one cluster size. By means of the *Smart-Vali-table* we can infer that the GVSM(TT) ($d_2 = 2$) reaches the optimal function at 92 cluster sizes of 99. LSA50 ($d_2 = 4$) accomplishes that at 5 cluster sizes and BVSM ($d_2 = 1$) 2 times. Further we can see that the *Vali-PP*-configuration $(d_1, d_2, d_3) = (3, 2, 3)$ appears at 50 of 99 cluster sizes and hence clearly dominates. The next best configurations are $(d_1, d_2, d_3) = (3, 2, 2)$ and $(d_1, d_2, d_3) = (3, 2, 1)$, what means that GVSM(TT) and meta configuration III is best in all three cases, representing 80% of all considered cluster sizes. Concerning the method ($d_3$) in these three cases, we have the optimality order: 3, 2, 1, i.e. HC, $k$-medoids, $k$-means. All dimension values in $d_1, d_2, d_3$ are enumerated in the same order as listed in Section 7.1.1, e.g. $d_2 = 2$ means GVSM(TT) and $d_1 = 1$ means meta Configuration I.

In a similar manner it can be concluded from the *Smart-Vali-tables* that LSA50 and HC is optimal in 89% of the cluster sizes under the Wemmert-Gancarski index, and LSA50 and HC is also optimal in 75% of the cluster sizes wrt. the Ray-Turi index. The meta configuration co-occurrence can be inferred from the *Smart-Vali-tables* in an analogous manner. In other words, *Smart-Vali-tables* permit statistical analysis of the co-occurrence distribution of the optimal *Vali-PP*-configurations. Concerning the YAML-1140 data set, only cluster sizes from 2 up to 100 were analyzed because the current QuantNet implementation requires cluster sizes in a range from 16 to 64. Thanks to the *Smart-Vali-plots*, one can easily see that it needs an initial period of cluster sizes (about 20) until the index functions start to consolidate their decreasing or increasing trend.

## 7.5 Vali-PP software and hardware infrastructure

In Figure 9 all software components are displayed which were integrated and developed in the course of the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$. They consist of various R functions, our R packages (in orange-blue), external R packages (in blue), and the *Smart-Vali-PP* component.

Particularly noteworthy features within the Vali-PP software infrastructure are the following. The object `optimal_share_index` in Listing 15 is basically a 3-dimensional table counting the number of intersections between every possible *Vali-PP*-configuration $d_1 \times d_2 \times d_3$ and the "optimal function". Due to the smart function `multi.which` from Listing 14, which was created by Mark van der Loo [21], all 3 *Smart-Vali-PP* functions `optimal_share.for.index`, `optimal_share.prettify` and `plot.optimal.functions` could be realized in a very compact and elegant way.

Since the *Vali-PP* calculations were time consuming, the whole process was parallelized over 24 or 32 CPU cores, depending on the used calculation server (Research Data Center, `https://rdc.hu-berlin.de`). The dimension "number of clusters" is the ideal quantity for massive parallelization. This allowed to complete the validation benchmark for the

[21] `https://github.com/markvanderloo`

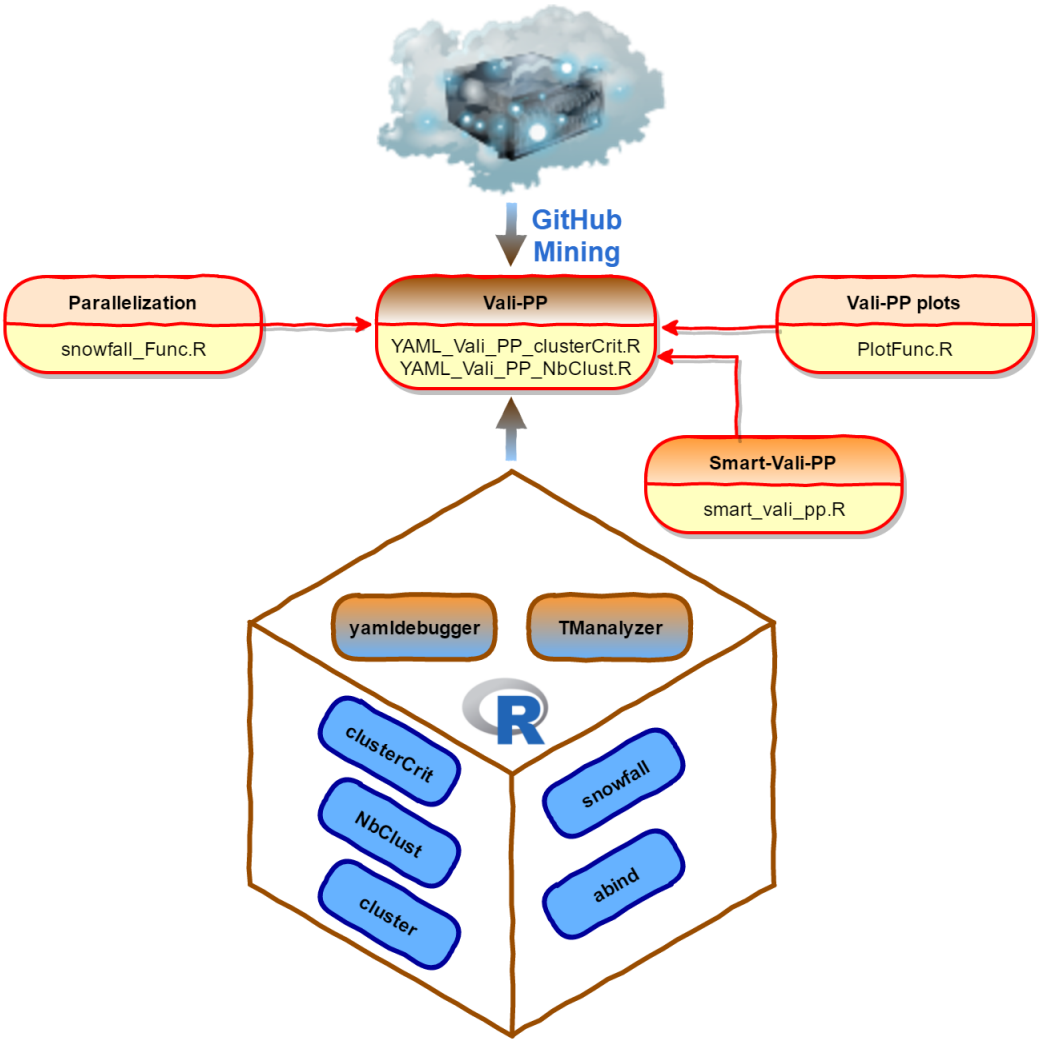


Figure 9: Software infrastructure of the Validation Pipeline components; blue: external R packages, orange-blue: our R packages; orange: our R functions;

YAML-500 data within one day for every package. In the case of **NbClust**, it took sometimes more than four hours to compute the results for a single index. The execution of **clusterCrit** ran faster, but still took approximately 10 hours for all 12 indices. The obvious reason for the better performance of **clusterCrit** is its C and Fortran 95 optimization [22].

Concerning YAML-1140, the *Vali-PP* benchmark was performed only for the **clusterCrit** package and cluster sizes up to 100. As discussed before, **clusterCrit** is better optimized, has a broader spectrum of quality indices and the upper limit of 100 clusters satisfies the practical needs. According to the results in Section "3.3 Benchmark" in Desgraupes (2013), the `intCriteria` function is quite efficient because the code is optimized to avoid duplicate calculations and to reuse values already computed for other indices. The function `clusterCritParallel` within the *Vali-PP* infrastructure was therefore adjusted. The vector of all selected indices is passed to the `intCriteria` function at once. Together with some other improvements like the calculation of the distance matrix in

[22] `https://github.com/cran/clusterCrit/tree/master/src`

advance (for the `pam` and `hclust` clustering functions), the following calculation times were measured, see Table 13. For better handling, the benchmark was executed for each meta configuration separately, the other four dimensions (*Vali-PP*-configurations) were processed in one pass. The full calculation for all meta configurations, and with it the entire *Vali-PP* benchmark, took around 2 hours and 20 minutes.

| Meta Configuration | time in seconds | physical/logical cores |
|---|---|---|
| I | 2410 | 24/24 |
| II | 2664 | 24/24 |
| III | 3364 | 12/24 |

Table 13: *Vali-PP* calculation time for YAML-1140 data, grouped by meta configurations

As the main objective of this section, we introduced and examined the so-called Validation Pipeline, a functional multi-staged instrument for clustering analysis, allowing multivariate statistical analysis of the distribution of driving factors (*Vali-PP*-configurations).

# 8 Discussion

## 8.1 Conclusion

The new package **rgithubS** enables a GitHub wide search for code and repositories using the GitHub Search API and allows to implement different parsers for data extraction from various software repositories retrieving smart data out of the raw text collection. Performing similar as Google, it is designed to find results that best meet the personal needs and which are ranked by best match, as indicated by the score field for each item returned. The QuantNet@GitHub statistics within **rgithubS** can be retrieved in real time. Due to some lightweight parsers of the package, basic mining tasks on QuantNet can be performed directly in the R console by use of **rgithubS** without further QuantNet Mining infrastructure R components.

The QuantNet Style Guide and the **yamldebugger** package allow a standardized audit and validation of YAML annotated software repositories. First, the **yamldebugger** checks the Quantlet repository structure, the validity of the YAML meta information and the completeness of the mandatory data fields according to the Style Guide. Second, it helps to analyze and unify the different YAML data fields, which are subject to varying spelling and notations. A meaningful and reasonable calibration of the matching list of the YAML data fields within the **yamldebugger** is crucial for further extraction of smart data within the TM and cluster validation steps.

The presented Google Analytics driven QuantNet Test Collection obtained from Google's metrics was used to evaluate and calibrate the IR performance. Three common text mining (TM) models were examined by means of the new **TManalyzer** package. By generating three performance measurement matrices: the number of retrieved documents, the number of retrieved and relevant documents (true positives) and the precision value for each query $\times$ TM model combination, the **TManalyzer** constitutes a convenient tool

for IR design and performance analysis. In this way, we can conclude that the number of true positives for all considered single term queries is maximal in the LSA50 model.

Further, we introduced the so-called **Validation Pipeline** (*Vali-PP*), a functional multi-staged instrument for clustering analysis, providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$. The *Smart-Vali-PP* framework, being a component of the overall *Vali-PP* infrastructure, allows to identify and quantify the clustering index optimality for a given Vali-PP-configuration. It can be applied to each of the 27 internal quality indices offered by the package **clusterCrit**. Considering the examined quality indices, we can conclude that the percentage of the optimality of the combination LSA50 and HC is in the range of 70-90% of all cluster sizes. In some cases the GVSM(TT) model reached this percentage level. We can also infer that the co-occurrence frequency of the meta configurations II and III is relatively high among the optimal Vali-PP-configurations, which indicates that more accurate and comprehensive metadata increases the clustering quality. Within the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$, the packages **yamldebugger** and **TManalyzer** were used to produce the meta information configurations and TM models.

## 8.2 Future Perspectives

The **TManalyzer** can be directly connected to the parser layer of the "GitHub API based QuantNet Mining infrastructure" and run as an R based search engine with three implicit TM models. Hence, the packages **rgithubS**, **yamldebugger** and **TManalyzer** are the necessary components for a self-contained GitHub mining engine in R. Due to the general approach, any set of text documents can serve as an object of text analysis. The **yamldebugger** acts as a smart data extraction layer and can be omitted or replaced by another text preprocessing component if another type of data is involved.

The new R packages presented in this paper build the integral components of the "GitHub API based QuantNet Mining infrastructure in R". The remaining components, such as the parser, clustering and D3 export layers, are available in experimental and working state. Very soon they will be implemented in a new R package, together with our research findings. Thus, the TM pipeline introduced in the beginning of this paper will be available in form of a package under the name "**tmPipelineQ**" and QuantNet's search engine called QuantNetXploRer will be finalized. The GitHub API driven QuantNetXploRer can be already found and used under `http://www.quantlet.de`.

> The psychological profiling [of a programmer] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large.
>
> *An interview with Donald Knuth. Dr. Dobb's Journal (April 1996)*

# References


Borke, L. (2017a). *TManalyzer: Provides IR tools in 3 text mining models: BVSM, GVSM(TT) and LSA*. R package version 0.5.0.
**URL:** *https://github.com/lborke/TManalyzer_dev*

Borke, L. (2017b). *yamldebugger: YAML parser debugger according to the QuantNet style guide*. R package version 1.0.
**URL:** *https://github.com/lborke/yamldebugger*

Borke, L. and Bykovskaya, S. (2017). GitHub Mining Infrastructure in R, *forthcoming* .

Borke, L. and Härdle, W. K. (2017). Q3-D3-LSA, *in* W. K. Härdle, H. H. Lu and X. Shen (eds), *Handbook of Big data Analytics*, Springer.

Bostock, M., Ogievetsky, V. and Heer, J. (2011). D3 Data-Driven Documents, *IEEE Transactions on Visualization and Computer Graphics* **17**(12): 2301–2309.
**URL:** *http://dx.doi.org/10.1109/TVCG.2011.185*

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008). clValid: An R Package for Cluster Validation, *Journal of Statistical Software* **25**(1): 1–22.
**URL:** *https://www.jstatsoft.org/index.php/jss/article/view/v025i04*

Charrad, M., Ghazzali, N., Boiteau, V. and Niknafs, A. (2014). NbClust: An R package for determining the relevant number of clusters in a data set, *Journal of Statistical Software* **61**(6): 1–36.
**URL:** *https://www.jstatsoft.org/article/view/v061i06*

Cleverdon, C. W. (1959). The evaluation of systems used in information retrieval (1958: Washington), *Proceedings of the International Conference on Scientific Information - Two Volumes*, National Research Council, Washington: National Academy of Sciences, pp. 687–698.

Cleverdon, C. W., Mills, J. and Keen, M. (1966). Factors determining the performance of indexing systems, *ASLIB Cranfield project, Cranfield*, ASLIB, pp. 37–59.
**URL:** *https://dspace.lib.cranfield.ac.uk/handle/1826/863*

Cristianini, N., Shawe-Taylor, J. and Lodhi, H. (2002). Latent Semantic Kernels, *Journal of Intelligent Information Systems* **18**(2): 127–152.
**URL:** *http://dx.doi.org/10.1023/A:1013625426931*

Desgraupes, B. (2013). *Clustering Indices*, University Paris Ouest, Lab Modal'X.

Desgraupes, B. (2016). *clusterCrit: Clustering Indices*. R package version 1.2.7.
**URL:** *https://CRAN.R-project.org/package=clusterCrit*

Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011). *Cluster Analysis*, John Wiley & Sons, Ltd.
**URL:** *http://dx.doi.org/10.1002/9780470977811.index*

Feinerer, I. and Hornik, K. (2015). *tm: Text Mining Package*. R package version 0.6-2.
**URL:** *http://CRAN.R-project.org/package=tm*

Feinerer, I., Hornik, K. and Meyer, D. (2008). Text Mining Infrastructure in R, *Journal of Statistical Software* **25**(5): 1–54.
**URL:** *http://www.jstatsoft.org/v25/i05/*

Feinerer, I. and Wild, F. (2007). Automated Coding of Qualitative Interviews with Latent Semantic Analysis, *in* Mayr and Karagiannis (eds), *Information Systems Technology and its Applications, 6th International Conference ISTA*, Gesellschaft für Informatik, Bonn, Germany, pp. 66–77.
**URL:** *http://nm.wu-wien.ac.at/research/publications/b679.pdf*

Gousios, G. and Spinellis, D. (2012). Ghtorrent: Github's data from a firehose, *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pp. 12–21.

Herbert, K. G., Wang, J. T. and Liu, J. (2004). Information Retrieval and Data Mining, *in* A. B. Tucker (ed.), *Computer Science Handbook, Second Edition*, 2nd edn, Chapman & Hall/CRC, pp. 75.1–75.5.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M. and Damian, D. (2014). The Promises and Perils of Mining GitHub, *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, ACM, New York, NY, USA, pp. 92–101.
**URL:** *http://doi.acm.org/10.1145/2597073.2597074*

Kaushik, A. (2010). *Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity*, Serious skills, Wiley.
**URL:** *https://books.google.de/books?id=nnZtAwAAQBAJ*

Lesk, M. and Salton, G. (1968). Relevance assessments and retrieval system evaluation, *Information Storage and Retrieval* **4**(4): 343 – 359.
**URL:** *http://www.sciencedirect.com/science/article/pii/0020027168900296*

Loeliger, J. (2009). *Version Control with Git - Powerful Tools and Techniques for Collaborative Software Development*, O'Reilly Media, Inc., Sebastopol.

Manning, C. D., Raghavan, P. and Schütze, H. (2008). *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.

North, S., Scheidegger, C., Urbanek, S. and Woodhull, G. (2015). Collaborative visual analysis with RCloud, *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pp. 25–32.

Pearmain, M., Mihailowski, N., Prajapati, V., Shah, K. and Remy, N. (2014). *RGoogleAnalytics: R Wrapper for the Google Analytics API*. R package version 0.1.1.
**URL:** *https://CRAN.R-project.org/package=RGoogleAnalytics*

Prem, E., Sanz, F. S., Lindorfer, M., Lampert, D. and Irran, J. (2016). Open Digital Science, *Technical report*, eutema GmbH (Austria) in co-operation with ZSI and Universidad de Zaragoza. available at https://www.researchgate.net/publication/303855957.

Rijsbergen, C. J. V. (1979). *Information Retrieval*, 2nd edn, Butterworth-Heinemann, Newton, MA, USA.

Salton, G. (1968). *Automatic Information Organization and Retrieval.*, McGraw Hill Text.

Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems, *Foundations and Trends®in Information Retrieval* **4**(4): 247–375.
**URL:** *http://dx.doi.org/10.1561/1500000009*

Scheidegger, C. (2016). *github: github API*. R package version 0.9.8.
**URL:** *https://github.com/cscheid/rgithub*

Scheidegger, C. and Borke, L. (2017). *rgithubS: GitHub API bindings for R - Special edition. Search, statistics, parsers*. R package version 0.9.9.
**URL:** *https://github.com/lborke/rgithubS*

Widgren, S. and others (2016). *git2r: Provides Access to Git Repositories*. R package version 0.14.0.
**URL:** *https://CRAN.R-project.org/package=git2r*

Wild, F. (2015). *lsa: Latent Semantic Analysis*. R package version 0.73.1.
**URL:** *https://CRAN.R-project.org/package=lsa*

Wild, F. and Stahl, C. (2007). Investigating Unstructured Texts with Latent Semantic Analysis, *in* R. Decker and H.-J. Lenz (eds), *Advances in Data Analysis. Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8-10, 2006*, Springer, Berlin Heidelberg, pp. 383–390.
**URL:** *http://www.springerlink.com/content/g7u377132gq5623g/*

Witten, I. H., Moffat, A. and Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition*, Morgan Kaufmann.
**URL:** *http://www.cs.mu.oz.au/mg/*

Xie, Y. (2016a). *formatR: Format R Code Automatically*. R package version 1.4.
**URL:** *https://CRAN.R-project.org/package=formatR*

Xie, Y. (2016b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.15.
**URL:** *http://yihui.name/knitr/*

# A Yamldebugger Application Example

```
> qnames = yaml.debugger.get.qnames(d_init$RootPath)
[1] "3 Q folder(s) found:"
[1] "ar1_process"  "random_walk"  "randomwalk_ar1"

> d_results = yaml.debugger.run(qnames, d_init)
[1] "1: ar1_process"
[1] "Simulates the path of a First-order autoregressive (AR-1) process over 50 ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - ar1_process.R"
[1] "Number of pictures: 2 - ar1_process-1.png, ar1_process-2.png"
[1] "-----------------------------------------------------------------------"
[1] "2: random_walk"
[1] "Simulates the path of a random walk over 50 time points. Epsilon terms ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - random_walk.R"
[1] "Number of pictures: 2 - random_walk-1.png, random_walk-2.png"
[1] "-----------------------------------------------------------------------"
[1] "3: randomwalk_ar1"
[1] "Similarity of both random walk and AR-1 (autoregressive process) to actual ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - randomwalk_ar1.R"
[1] "Number of pictures: 6 - randomwalk_ar1_0.8_1.png, randomwalk_ar1_0.8_2.png,
    randomwalk_ar1_0.8_3.png, randomwalk_ar1_0.95_1.png, randomwalk_ar1_0.95_2.png,
    randomwalk_ar1_0.95_3.png"
[1] "-----------------------------------------------------------------------"

> ( OverView = yaml.debugger.summary(qnames, d_results, summaryType = "mini") )
  Q-Quali Q folders Q Names  Descriptions stats          Keywords stats
1 A ar1_process  ar1_process 32 word(s), 157 Character(s) 9: 9 (standard), 0 (new)
2 A random_walk  random_walk 17 word(s), 93 Character(s) 9: 9 (standard), 0 (new)
3 A randomwalk_ar1 randomwalk_ar1 58 word(s), 273 Character(s) 12: 12 (standard), ...
```

Listing 11: **yamldebugger** application example

# B Example for YAML data field analysis

```
> subset(OverView, !(`Q-Quali` %in% c("A")) )
[1] Q-Quali  Q folders  Q Names  Descriptions stats Keywords stats
<0 rows>

> as.data.frame(d_results$meta_names_distribution)
               d_results$meta_names_distribution
Author                                         3
Description                                    3
Example                                        3
Input                                          3
Keywords                                       3
Name of Quantlet                               3
Published in                                   3
See also                                       3
Submitted                                      3

> yaml.not.Qdfields(d_results$meta_names_distribution)
character(0)

> sapply( d_results$Metainfos, function(yaml){ yaml.Qdfields.nchar.from.meta(yaml) } )
   [,1] [,2] [,3]
q    11   11   14
p    20   20   20
a    11   11   11
d   222  117  370
k   137  134  177
df    0    0    0
e    82   44  417
i    85   53   85
o     0    0    0
s    31   31   32
sa   27   27   24
ce    0    0    0
cp    0    0    0
cw    0    0    0
od    0    0    0
sf    0    0    0
u     0    0    0

> rowSums(sapply( d_results$Metainfos, function(yaml){ yaml.Qdfields.nchar.from.meta(
    yaml) } ))
  q   p   a   d   k  df   e   i   o   s  sa  ce  cp  cw  od  sf   u
 36  60  33 709 448   0 543 223   0  94  78   0   0   0   0   0   0

> d_names = unlist(sapply( d_results$Metainfos, function(yaml){
    yaml.Qdfields.from.meta(yaml)$found_dnames } ))
> ( d_names_distr = sort(table(d_names), decreasing = TRUE) )
d_names
 a  d  e  i  k  p  q  s sa
 3  3  3  3  3  3  3  3  3
```

Listing 12: Example for YAML data field analysis via **yamldebugger** functions based on the results from Listing 11

# C Word clouds of YAML keywords

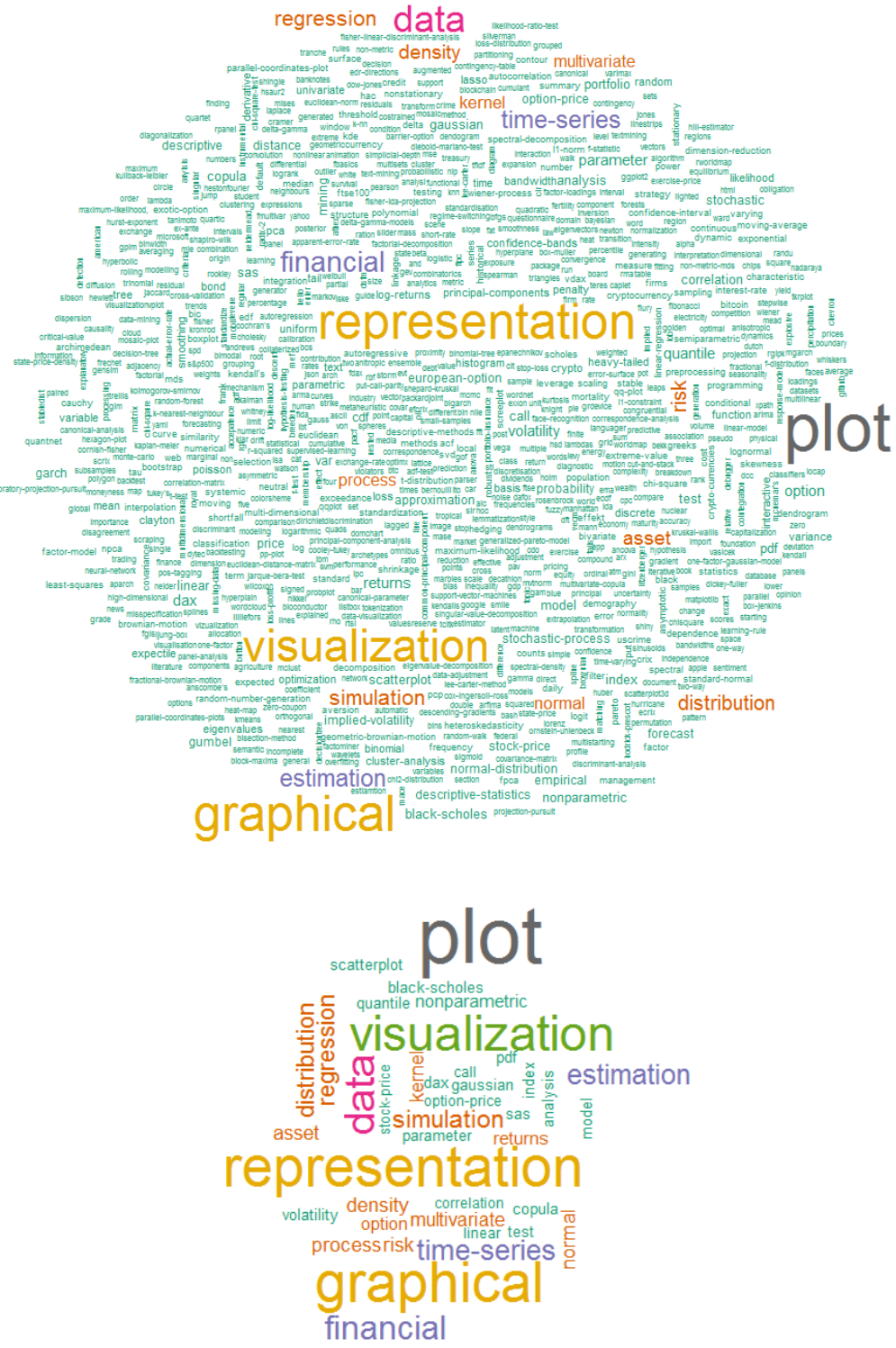


Figure 10: Word clouds of the keywords extracted from the Quantlets' YAML meta info

The word clouds were created by yaml_wordcloud.

# D TManalyzer application example

```
library(yamldebugger)
library(TManalyzer)
(obj.names = load("yaml_list_full_20161122.RData", .GlobalEnv))  # 1140 Docs

t_vec = yaml.list.extract(yaml_list, weight = c(q=1, d=1, k=1, p=1))
# tf_weight = "nnc" for tf weighting; tf_weight = "ntc" for tf-idf weighting
system.time( tm_list <- tm.create.models(t_vec, tf_weight = "nnc") )
[1] "Dim TDM: 1211,1140"
[1] "Dim LSA Auto: 154"
      User     System Elapsed
     12.47       0.06      12.58

# Single term queries
query = c("covar", "random", "quantile", "histogram", "multivariate")
# Compound term queries
query = c("random number", "multivariate statistics", "black scholes")

query.tm.folded = query.tm.fold_in(query, tm_list, tf_weight = "nnc")
q_tdm_sim.tm_res = q_tdm_sim.tm.list(query.tm.folded)

# 3 threshold levels for IR
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8)
(m1 = q_ir_list$retrieved_m)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.7)
(m2 = q_ir_list$retrieved_m)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.6)
(m3 = q_ir_list$retrieved_m)
colnames(m1) = colnames(m2) = colnames(m3) = c("B", "TT", "LSA", "L50")
( m_full = cbind(m1, m2, m3) )
              B TT LSA L50 B TT LSA L50 B TT LSA L50
covar         0  0   0   7 0  0   3   9 0  0   6   9
random        0  0   0   6 0  1   0   7 0 10   3  18
quantile      0  0   0   0 0  1   0   1 0  1   2   6
histogram     0  0   0   3 0  0   2   6 2  2   4  14
multivariate  0  0   0   0 0  0   0   4 0  0   0  16

query.tm.folded$q_tdm
          Docs
  Terms    q1 q2 q3 q4 q5
  covar     1  0  0  0  0
  histogram 0  0  0  1  0
  multivari 0  0  0  0  1
  quantil   0  0  1  0  0
  random    0  1  0  0  0
```

Listing 13: IR system designs via **TManalyzer**

# E Smart-Vali-PP application example

```
# A which for multidimensional arrays. Mark van der Loo 16.09.2011
# A Array of booleans
# returns a sum(A) x length(dim(A)) array of multi-indices where A == TRUE
multi.which <- function(A){
    if ( is.vector(A) ) return(which(A))
    d <- dim(A)
    T <- which(A) - 1
    nd <- length(d)
    t( sapply(T, function(t){
        I <- integer(nd)
        I[1] <- t %% d[1]
        sapply(2:nd, function(j){
            I[j] <<- (t %/% prod(d[1:(j-1)])) %% d[j]
        })
        I
    }) + 1 )
}
```

Listing 14: *multi.which* for multidimensional arrays

```
library(clusterCrit)
source("smart_vali_pp.R")
(load("results/obj_pp_d_clusterCrit_1140Q_20161222.RData", .GlobalEnv))

(ind_names = getCriteriaNames(TRUE)[c(1,3,4,5,7,28,31,32,37,39,41,42)])
# [1] "Ball_Hall"       "C_index"         "Calinski_Harabasz" "Davies_Bouldin"
# [5] "Dunn"            "McClain_Rao"     "Ray_Turi"          "Ratkowsky_Lance"
# [9] "Silhouette"      "Trace_W"         "Wemmert_Gancarski" "Xie_Beni"

# only max/min interpretation
ind_goal_func = c(min, min, max, min, max, min, min, max, max, min, max, min)

i = 2 ; yrange = c(0, 0.4) # clusterCrit//C_index
i = 7 ; yrange = c(0, 12)  # clusterCrit//Ray_Turi
i = 11; yrange = c(0, 0.5) # clusterCrit//Wemmert_Gancarski

# 4-dim array for the selected index i : Meta conf / Methods / Models / nclusters
sel_index_vali = pp_d[i,,,,]
index.optimal.func = apply(sel_index_vali,
                           length(dim(sel_index_vali)), ind_goal_func[[i]])
optimal_share_index = optimal_share.for.index(sel_index_vali, index.optimal.func)

# Main Evaluation 1
optimal_share.prettify(optimal_share_index)
# Main Evaluation 2
# plot only graphs which intersect the best function at least at one point
plot.optimal.functions(sel_index_vali, index.optimal.func, optimal_share_index,
                       yrange, ind_names[i])
# plot ALL graphs of Vali-PP configurations
plot.optimal.functions(sel_index_vali, index.optimal.func, optimal_share_index,
                       yrange, ind_names[i], only_partly_best = FALSE)
```

Listing 15: **Smart-Vali-PP** application example

# SFB 649 Discussion Paper Series 2017

For a complete list of Discussion Papers published by the SFB 649, please visit http://sfb649.wiwi.hu-berlin.de.

001 "Fake Alpha" by Marcel Müller, Tobias Rosenberger and Marliese Uhrig-Homburg, January 2017.
002 "Estimating location values of agricultural land" by Georg Helbing, Zhiwei Shen, Martin Odening and Matthias Ritter, January 2017.
003 "FRM: a Financial Risk Meter based on penalizing tail events occurrence" by Lining Yu, Wolfgang Karl Härdle, Lukas Borke and Thijs Benschop, January 2017.
004 "Tail event driven networks of SIFIs" by Cathy Yi-Hsuan Chen, Wolfgang Karl Härdle and Yarema Okhrin, January 2017.
005 "Dynamic Valuation of Weather Derivatives under Default Risk" by Wolfgang Karl Härdle and Maria Osipenko, February 2017.
006 "RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods" by Lukas Borke, February 2017.
007 "Testing Missing at Random using Instrumental Variables" by Christoph Breunig, February 2017.
008 "GitHub API based QuantNet Mining infrastructure in R" by Lukas Borke and Wolfgang K. Härdle, February 2017.

**SFB 649, Spandauer Straße 1, D-10178 Berlin**
**http://sfb649.wiwi.hu-berlin.de**

This research was supported by the Deutsche Forschungsgemeinschaft through the SFB 649 "Economic Risk".